COMBINING IMAGE PROCESSING WITH SIGNAL PROCESSING TO IMPROVE TRANSMITTER GEOLOCATION ESTIMATION

THESIS

Amy M. Abraham, Second Lieutenant, USAF

AFIT-ENG-14-M-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-14-M-01

COMBINING IMAGE PROCESSING WITH SIGNAL PROCESSING TO IMPROVE TRANSMITTER GEOLOCATION ESTIMATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Amy M. Abraham, B.S.E.E.

Second Lieutenant, USAF

March 2014

AFIT-ENG-14-M-01

COMBINING IMAGE PROCESSING WITH SIGNAL PROCESSING TO IMPROVE TRANSMITTER GEOLOCATION ESTIMATION

Amy M. Abraham, B.S.E.E.
Second Lieutenant, USAF

Approved:

//signed//	25 Feb 2014
Richard K. Martin, PhD (Chairman)	Date
//signed//	25 Feb 2014
Lt Col Jeffrey D. Clark, PhD (Member)	Date
//signed//	25 Feb 2014
Michael A. Temple, PhD (Member)	Date

AFIT-ENG-14-M-01

Abstract

This research develops an algorithm which combines image processing with signal processing to improve transmitter geolocation capability. A building extraction algorithm is compiled from current techniques in order to provide the locations of rectangular buildings within an aerial, orthorectified, RGB image to a geolocation algorithm. The geolocation algorithm relies on measured time difference of arrival (TDOA) data from multiple ground sensors to locate a transmitter by searching a grid of possible transmitter locations within the image region. At each evaluated grid point, theoretical TDOA values are computed for comparison to the measured TDOA values. To compute the theoretical values, the shortest path length between the transmitter and each of the sensors is determined. The building locations are used to determine if the line of sight (LOS) path between these two points is obstructed and what would be the shortest reflected path length. The grid location producing theoretical TDOA values closest to the measured TDOA values is the result of the algorithm. Measured TDOA data is simulated in this thesis. The thesis method performance is compared to that of a current geolocation method that uses Taylor series expansion to solve for the intersection of hyperbolic curves created by the TDOA data. The average online runtime of thesis simulations range from around 20 seconds to around 2 minutes, while the Taylor series method only takes about 0.02 seconds. The thesis method also includes an offline runtime of up to 30 minutes for a given image region and sensor configuration. The thesis method improves transmitter geolocation error by an average of 44m, or 53% in the obstructed simulation cases when compared with the current Taylor series method. However, in cases when all sensors have a direct LOS, the current method performs more accurately. Therefore, the thesis method is most applicable to missions requiring tracking of slower-moving targets in an urban environment with stationary sensors.

Thank you to all my friends and family who knew when to distract me from my thesis and when to make me work.

Table of Contents

Page

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
AFIT	Air Force Institute of Technology
AWGN	additive white Gaussian noise
BW	black and white
CDF	cumulative distribution function
FFT	fast Fourier transform
GSD	ground sample distance
LOS	line of sight
LS	least-squares
MATLAB®	matrix laboratory
MLE	maximum likelihood estimator
NTSC	National Television Standards Commission
PDF	probability density function
RF	radio frequency
SI	spherical interpolation
TDOA	time difference of arrival
TOA	time of arrival

COMBINING IMAGE PROCESSING WITH
SIGNAL PROCESSING TO IMPROVE
TRANSMITTER GEOLOCATION ESTIMATION

I. Introduction

This chapter provides an overview of the contents of this thesis. The chapter discusses why this research area is significant and explains the approach taken.

1.1 Research Motivation and Related Research

There are many situations when it is necessary or useful to locate someone or something by using the signal emissions from a radio transmitter on the ground. Geolocation can be applied to civilian, military, commercial, government, domestic, or overseas situations. The applications are boundless. Due to the prevalent need for this capability, there are many methods already developed for accomplishing geolocation. Most, if not all, of these methods rely on signal timing information and, more specifically, on time difference of arrival (TDOA) data. When a sensor detects a signal, usually the sensor has no way of knowing when the signal was transmitted and therefore no way of knowing the travel time, or time of arrival (TOA), of the signal. There are rare situations when the TOA can be estimated. For example, [10] estimates the TOA of a prompted signal transmitted from a cell phone using the round trip time calculated as the time between when the signal was prompted and when it was received. However, collecting TOA in this manner requires logistical and universal planning, which is not always practical or available. The TDOA data is therefore much more commonly used. Though TOA can rarely be estimated, relative timing can be calculated if there are multiple

sensors. The first sensor to detect the signal must be closest to the transmitter, since the time it takes for a radio signal to travel is directly proportional to the distance it travels. In this way, TDOA data can narrow down the location of the transmitter. If the number of sensors is at least one greater than the number of unknown coordinates, then this location can be determined precisely. This is because the number of TDOA estimates is one less than the number of sensors.

There are many techniques which have been developed to use TDOA values from multiple sensors for position localization. Some of them only apply to spatially linear sensors, such as beamforming or those assuming a distant source [3]. More generalized techniques which can apply to any sensor configuration are more complex. These techniques solve for the intersection of a set of hyperbolic curves which are defined by the TDOA data. Of all these methods, few provide a precise estimate under noisy conditions or make use of extra sensor information if there are more than the required number of sensors. One of these few methods, which is commonly used, is the Taylor series method [3, 10]. It is an iterative method which improves upon an initial location estimate by determining the local least-squares (LS) solution [3, 10]. It continues to improve the solution until the difference in the improvements becomes smaller than a given threshold or until a given number of iterations have been performed. If the initial location guess is not close enough to the actual location, then local minima may be mistaken for the actual location. Also, the solution may not converge. Aside from these issues, the Taylor series method provides very accurate results under noisy conditions when there are no obstructions between the sensors and the transmitter. Therefore, this is the method that will be compared against the method developed in this thesis.

The method developed in this thesis seeks to improve geolocation estimation when there are obstructions between the sensors and the transmitter. This is a relatively novel exploration. There is a method discussed by [10] which attempts to correct the bias to the

TDOA values caused by the longer path taken by the signal in the obstructed case, but this method relies on assumptions about the variance and distribution of the biased data, rather than on knowledge about the actual obstructions [10]. The only localization methods found to incorporate predictability using the actual environment are based on small, easily controlled, indoor environments and do not involve obstructions. Reference [12] presents a method for indoor speaker localization by using the TDOA of sound waves, which behave similarly to radio waves. In an indoor environment, TDOA data is corrupted by reverberations from the enclosing walls. Reference [12] uses conditional probability density functions created from training data taken by placing the speaker at representative locations in the room in order to classify the experimental data into angular regions of the room. Even if the solution for dealing with reverberations proposed by [12] could be applied to obstructions, and if classification by region were possible on the much larger scale of an outdoor urban environment, the extent of the required prior knowledge is impractical. Therefore, this thesis proposes a new method of accounting for non-line of sight (LOS) situations in geolocation, where the only prior knowledge of the environment that is required is an aerial, orthorectified, RGB image of the area in which the transmitter and the sensors are located.

The aerial image can be used to find the locations of buildings which may obstruct the LOS between a transmitter and a sensor. Fortunately, the need for image processing is just as prevalent as it is for geolocation. There are many building extraction algorithms already developed. The algorithm developed in this thesis is merely a compilation of numerous current techniques already in use. The purpose of creating the building extraction algorithm in this thesis is simply to provide realistic data to the geolocation algorithm and to show how this data was obtained. Any building extraction algorithm can be used in place of the one developed here as long as it provides the required data to the geolocation algorithm.

1.2 Research Goal

The goal of this research was to improve upon existing geolocation methods of locating a transmitter using timing information from multiple sensors. This was accomplished by applying knowledge gained from using existing methods to process an aerial image of the environment in which the transmitter and the sensors are located. Current geolocation methods either do not consider or do not accurately correct the corruption of signal timing information caused by reflections. Extracting the locations of buildings from aerial images provides location information about the reflecting and obstructing surfaces which affect the signal timing information. Information about the non-LOS signal paths can greatly improve the accuracy of geolocation.

1.3 Research Methodology

The method proposed by this thesis begins with an orthorectified, aerial, RGB image. All image processing techniques used to extract building locations existed prior to this research. Along with the image, two other inputs to the building extraction algorithm are required, including a rough minimum expected building perimeter estimate and a rough maximum expected building perimeter estimate. The RGB values in the image are then converted to grayscale intensity values and a contrast enhancement technique is applied. The pixels with an intensity below a certain threshold are classified as shadows, and the remaining pixels in the image are clustered based on their intensity values. The connected pixels in each cluster are potential building objects. The shadow objects are processed separately and classified based on their shape and size. A direction is calculated describing which side of the shadows is likely to overlap the building that casts the shadow. Then, each shadow is compared to each building object. Those building objects which overlap the same shadow on the correct side of that shadow are considered part of the same object. Each of these objects is then classified based on size and shape. The remaining objects are

considered to be the buildings in the image, and a rectangle of best fit is found for each object. It is assumed for simplicity that the buildings are all rectangular with four walls.

Since TDOA data is not available for the overhead images used in this thesis, the TDOA data is simulated. Transmitter and sensor locations are arbitrarily chosen, and the shortest path from the transmitter to each of the sensors is found using the locations of the buidings along with their walls. It is assumed that all buildings walls are high enough intercept the signal and that all sensors as well as the transmitter are high enough off the ground that the terrain does not intercept the signal. It is also assumed that the wall surfaces of the buildings are rough enough to exhibit Lambertian reflectance. This means that when a radio wave hits a wall, it is diffused in all directions from that side of the wall. Therefore, every point on the surface of a wall is considered a valid reflection point, whereby the reflection is always received by a sensor as long as LOS is not obstructed. Finally, it is assumed that a path is not viable if there are more than two reflections in the path, as this would diminish the signal power to an undetectable level. With these assumptions, a list of walls facing the transmitter with a direct LOS is created. These walls are potential reflection points between the transmitter and the sensors. From there, a second list is created including the walls from the first list and any walls which face each of them with a direct LOS. The wall pairings in the second list are potential points of reflection in a double-reflection path between the transmitter and the sensors. To complete the paths, the sensor locations are compared to the open walls in each list. The lengths of the completed paths are then compared to find the shortest distance from the transmitter to each sensor. These pixel distances are then converted to time values using the propagation speed of radio waves and the ground sample distance of the image. These time values represent TOA values at each sensor, and after noise is added, they are used to find the TDOA values for each sensor.

Once the actual transmitter location is chosen, and the corresponding TDOA data are simulated, a grid search is peformed to determine the location of the transmitter. At each location in the grid search, the same steps that were performed to simulate the TDOA data are performed to create the TDOA data which would theoretically be measured at the sensors if that grid location were the transmitter location. The theoretical TDOA data are compared against the simulated TDOA data, and the grid location which creates the most similar data is determined to be the location of the transmitter.

1.4 Thesis Organization

Chapter II presents and examines relevant existing concepts and methods both for image processing and for geolocation. The chapter presents the methods for image processing before discussing the current concepts of geolocation. It steps through ideas for preprocessing, for segmentation into buildings, and for post-processing. It is evident that there are many successful building extraction methods currently available. Then, the chapter discusses a few methods of geolocation which assume LOS conditions. This discussion includes the Taylor series method of evaluating the hyperbolic curves created by TDOA data. The Taylor series method is used as a baseline against which to compare the improved method proposed in this thesis. The grid search method of finding the maximum likelihood estimator (MLE) for a transmitter location is also discussed. This is not an original technique. The chapter ends with a description of an existing concept for determining the shortest path between two points when the direct LOS is not an option.

In *Chapter III*, the steps taken to develop the algorithm proposed by this thesis are explained from image processing for building locations to geolocating a transmitter using multiple sensors. None of the techniques used are original. However, the manner in which the image information is combined with the signal information is a novel concept.

Chapter IV illustrates and explicates the results of the algorithm in various situations compared with the competing Taylor series method. The building extraction portion of the

algorithm is applied to five different orthorectified, aerial, RGB images. The remaining geolocation portion of the algorithm is applied to the extracted buildings in three of these images. Furthermore, the results of five different additive white Gaussian noise (AWGN) standard deviation (σ_N) values, five different transmitter locations, and five different sensor configurations are compared between the thesis method and the Taylor series method for both runtime and error.

Finally, *Chapter V* discusses the implications of the results and the usefulness of the developed algorithm. Areas for further improvement and exploration are discussed as well.

II. Literature Review

This chapter describes existing concepts and methods for obtaining the desired information in this field of interest. The imaging side of the field is examined first in this chapter. The goal of image processing in this thesis is to extract the two-dimensional locations of any buildings present in the image plane. In order to accomplish this goal, Section 2.1 first discusses possible modifications to color images which can be helpful to the subsequent extraction techniques. Then, Section 2.2 details various methods of breaking down an image into separate regions. Following this image segmentation, Section 2.3 introduces ways to deal with the challenges of classifying the appropriate segmented regions of an image as actual buildings. Following the discussion on image processing with the purpose of building extraction, this chapter describes methods of signal processing with the purpose of geolocation. Section 2.4 approaches the basic problem of how to use signal timing information received from multiple sensors with unobstructed lines of sight to a signal source in order to locate that source in a two-dimensional plane. Then, in Section 2.5 the problem is expanded to the obstructed case. The chapter ends with a proposal of how building extraction can be used to improve geolocation accuracy in these cases.

2.1 Image Preprocessing Methods

Processing an image before analyzing it for objects of interest can improve efficiency and provide more accurate results in the long run. This section offers a few ideas which can aid in image processing as well as methods for implementing those ideas.

2.1.1 Color Models.

In this thesis, colors are not dealt with in the traditional sense. Rather, the color components of an image are converted to a single value to enable direct comparison of

pixel color content within the image. This conversion can be accomplished in a number of ways, but all of the methods discussed in this subsection begin with an RGB color model representation of an image. This means that each pixel in the original image has three values assigned to it. One value describes the red channel content in the pixel, the next value describes the green channel content, and the third describes the blue channel content. Combining these three values into a single value is a necessary preprocessing step for all of the color segmentation methods discussed in this chapter, and the following sections assume this step has been performed when referring to the color value of a pixel or region.

All of the color models and corresponding channel equations discussed in this subsection are derived from [13]. After the RGB color model, the most common is the YIQ model, described by Equation (2.1). This is the standard model used in National Television Standards Commission (NTSC) color TV transmission. The Y channel roughly corresponds to luminance or intensity, and the I and Q channels correspond to chroma, or hue and saturation. Equation (2.1) shows that the Y channel transformation is most heavily weighted towards the green channel component. This is because the human eye registers green color more easily than it does red or blue color, which explains why this model is the standard in television.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.1}$$

The HSI model is another color model option, and Equation (2.2) describes the channel transformations from the RGB model to this model.

$$\begin{bmatrix} I \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{-\sqrt{6}}{6} & \frac{-\sqrt{6}}{6} & \frac{\sqrt{6}}{3} \\ \frac{1}{\sqrt{6}} & \frac{-2}{\sqrt{6}} & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.2a}$$

$$S = \sqrt{V_1^2 + V_2^2} \tag{2.2b}$$

$$H = \tan^{-1}\left(\frac{V_2}{V_1}\right) \text{ if } V_1 \neq 0, \text{ otherwise } H \text{ is undefined.} \tag{2.2c}$$

The H channel corresponds to hue, the S channel corresponds to saturation, and the I channel corresponds to intensity, or brightness. Some other color models are HSV and HCV. In both the HSV and the HCV color models, the V channel is the intensity channel, and the transformation is exactly the same as the one used for the intensity I channel in Equation (2.2). When analyzing image content for specific objects, it makes sense to equally weight RGB channels. This paper utilizes the intensity channel calculated from the transformation used in all three of these models to compare pixel values in an image.

2.1.2 Normalization, Thresholding, and Contrast Enhancement.

For some methods of image analysis, it may be beneficial to normalize the pixel values of an image. If an image has been converted to grayscale or to another color model channel as discussed above, then pixel values in the image will range from 0 to 255. Reference [2] chooses to normalize the pixel values by interpolating the grayscale values in a range from 0 to 1. They found the normalized values easier to understand and manipulate. However, the merit of normalization really just depends on the input range expected by the programs and functions being implemented.

Another way to simplify pixel values in an image is called thresholding. Pixel values below a certain threshold value are assumed to be part of the background and are all set to the same value, and the pixels above the threshold value are assumed to be part of the foreground and are all set equal to each other as well [2]. In this way, the image is converted to a binary image. Thresholding as a preprocessing step can be very useful

when edge detection is used for image segmentation, since it decreases noise and isolates regions of interest. This simplification step may result in an undesirable loss of data, however, if the image is complex. Edge detection image segmentation can also benefit from boundary modification as a preprocessing step. This process simply changes the value of all pixels which lie along the boundary of the image to a background pixel value [2]. This way, any objects which lie on the border of the image are artificially given a detectable edge, which aids in finding close object boundaries [2].

Another pixel value manipulation that can be helpful in the preprocessing stages is contrast enhancement. Contrast enhancement can make edges and other variations more easily detected. There are multiple ways of enhancing the contrast in an image. One way is through histogram equalization, which is accomplished by spreading the pixel values in the image evenly across the entire possible range of values. The result of applying histogram equalization to an image is a probability density function (PDF) of pixel values that looks like a uniform distribution and a cumulative distribution function (CDF) that looks like a line with a slope of one. Another method of contrast enhancement also involves spreading the pixel values across the full possible range of values. The difference is that the pixels are mapped to the new values, which only stretches the original distribution, rather than equalizing the histogram. This technique makes the existing variation in the image more obvious without introducing new variation. An example of contrast enhancement, which was performed by [2], is shown in Figure 2.1. The PDF of the original image is weighted toward the brighter end of the range of values. After contrast enhancement, the image brightness has been toned down to reveal variations that were less visible in the original image.

2.1.3 Filtering.

The descriptions of filtering benefits and techniques in this subsection are taken from [2]. Filtering is a way of removing noise or unwanted artifacts from an image, but it can

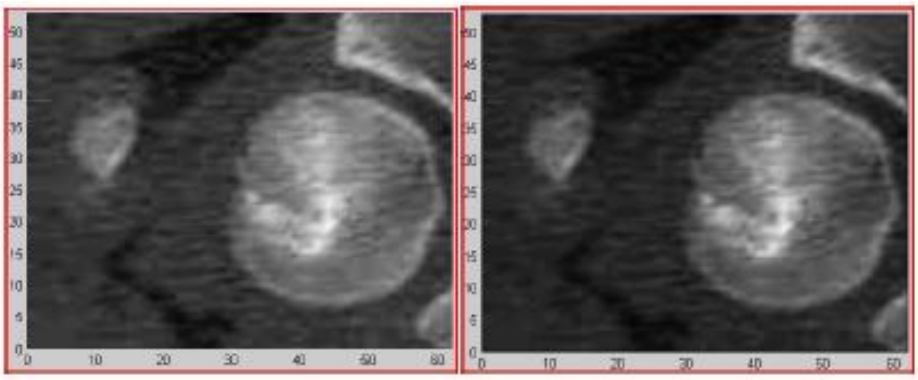

Figure 2.1: An example of an image (a) before and (b) after contrast enhancement [2]. Used with permission.

sometimes be tricky to accomplish this removal without also removing important information from the image. Reference [2] compares various filters in Figure 2.2. Reference [2] found the median filter to perform the best for edge detection purposes, since it removes noisy salt and peppering while maintaining edges.

2.2 Image Segmentation Methods

This section covers research which explores various methods of partitioning an image in order to analyze certain groups of pixels. The goal of each method is to create pixel groups describing individual objects which have the potential of representing a building. Many of the methods that will be described rely on color values for segmentation. After segmentation based on color, each segmented region is usually a group of relatively homogeneous pixels. This means they were grouped together to represent an object under the assumption that objects of interest are monochromatic. These methods therefore require post-processing to determine not only which regions represent objects of interest (potential buildings), but also which regions may represent parts of the same

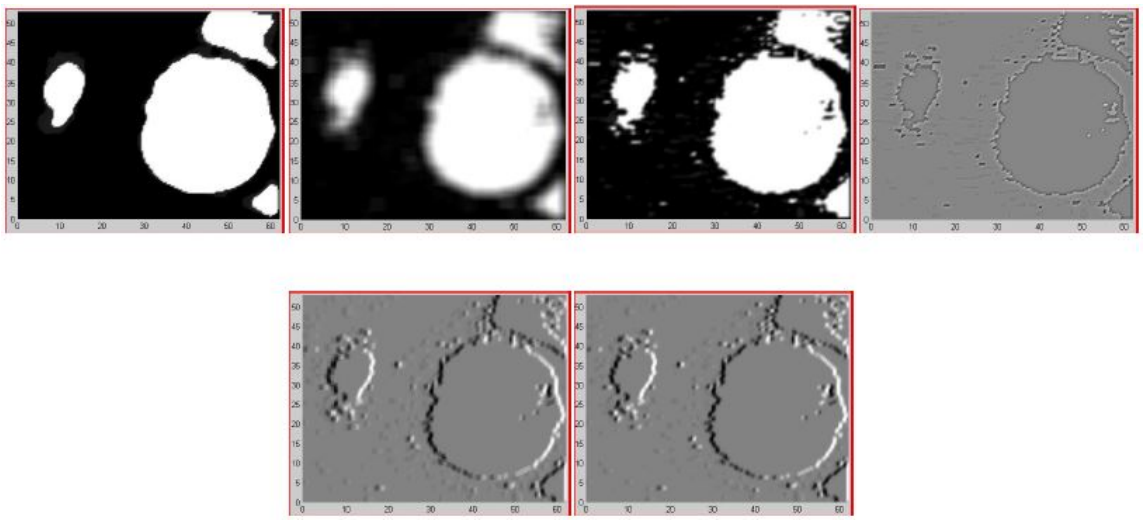

Figure 2.2: An example of a medical CT image after (left to right) (a) median filtering; (b) averaging filtering; (c) Gaussian low pass filtering; (d) Laplacian filtering; (e) Prewitt filtering; and (f) Sobel filtering [2]. Used with permission

polychromatic object. Relevant post-processing techniques will be discussed in Section 2.3.

2.2.1 Edge Detection.

Unless otherwise specified, the information included in this subsection is extracted from [2] and focuses on edge detection as a form of image segmentation. Reference [2] applies segmentation methods to a medical CT scan, rather than an overhead image, but the concepts apply to either type of image. Edge detection is a fairly straightforward concept, and there are many techniques used to accomplish the task. Most techniques find edges by defining them as areas of abrupt change from one group of similar color values to another group of similar color values. Reference [2] claims that the Canny edge detection algorithm provides the best results when compared with the Prewitt, the Sobel operator, and the Hough transform, measured in terms of detecting the edges, locating them accurately, and marking them uniquely. However, even the best edge detection

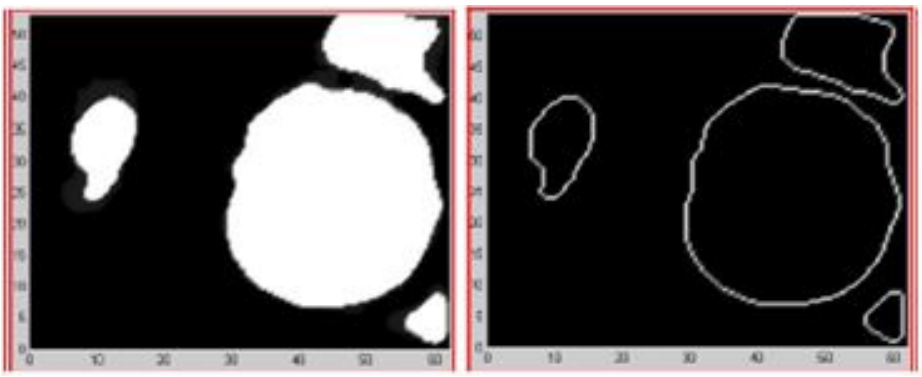

Figure 2.3: An example of an image (a) before and (b) after Canny edge detection [2]. Used with permission.

methods are very susceptible to noise, and their reliance on classifying levels of variation in a region as either part of the same object or the start of a new object can lead to false edges or gaps, rather than accurate and closed object boundaries. Therefore, preprocessing can be very helpful, and post-processing of the edges is almost always required to accomplish segmentation. This is why [2] also employs the Moore neighborhood boundary tracing technique following the Canny edge detection.

Figure 2.3 shows the results of applying Canny edge detection to an image. Before applying this detection, however, [2] chooses to include thresholding as one of the preprocessing steps following a conversion to grayscale values. This is why the image shown in Figure 2.3 is a binary image and also why the edge detection is able to produce such clean results.

Then, [2] uses the Moore neighborhood boundary tracing technique to locate pixels along the detected edges and connect them. Any pixels residing inside the boundary are converted to the value of the other pixels in the region. Another step used by [2] to complete the image segmentatio based on edge detection is referred to as “point in polygon.” In this step, the contour resulting from the Moore tracing step is the polygon,

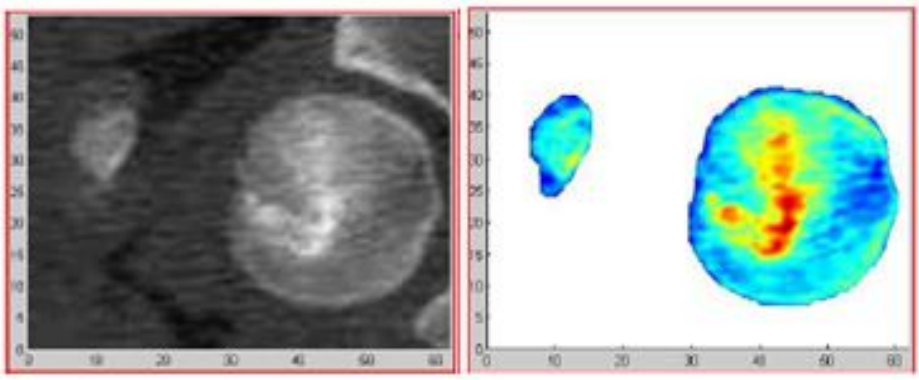

Figure 2.4: An example of an (a) original image alongside (b) the results of edge detection segmentation [2]. Used with permission.

and the pixels in the original image are points that are tested to see whether they lie within the polygon. The results are shown in Figure 2.4.

Canny edge detection is used by [6] as well, but it is applied to a more complex overhead image similar to the one in Figure 2.5, rather than to a medical CT scan. Figure 2.5 shows how inconclusive edge detection on a more complex image can be. Therefore, just as [2] used Moore neighborhood boundary tracing to make sense of the detected edges, so does [6] use a technique they call USC LINEAR linking approximation. This technique combines parallel edges that are very close to one another into a single edge and groups other edges to reduce the presence of fragments [6]. Further post-processing of image segmentation will be discussed in Section 2.3.

2.2.2 Seeded Region Growing.

All theory in this subsection is derived from information found in [8]. The seeded region growing method of image segmentation groups pixels based on one feature, usually color. For the seed points, the algorithm chooses singular pixels spaced at regular intervals throughout an image. The size of these intervals corresponds to the minimum expected size in pixels of a building in the image. If the seed pixels are spaced too far apart, the

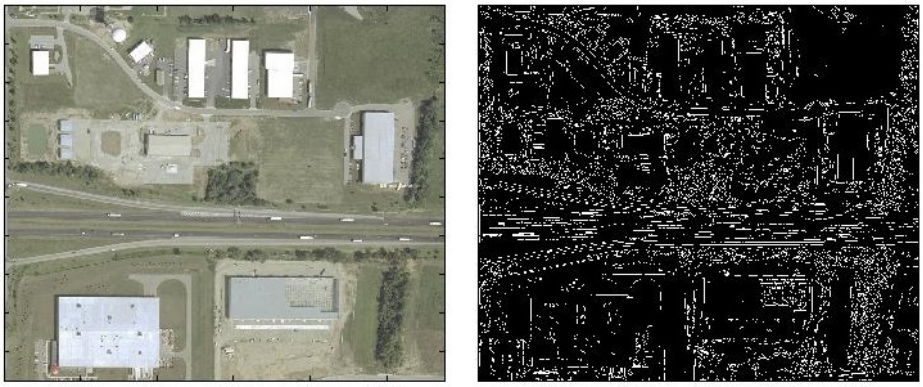

Figure 2.5: An example of an (a) original overhead image alongside (b) the results of Canny edge detection with a threshold determined automatically in MATLAB®. Original image data available from the U.S. Geological Survey [1].

algorithm could mistakenly group multiple buildings into one object. A modified method of choosing the seed points entails dividing the image into equal initial regions, and then arbitrarily choosing one pixel in each region that resembles a building pixel. In [8], seed points were chosen using this modified method and basing the resemblance on an assumption that the roofs of buildings are most likely to be red or gray in color. Therefore, the seed pixel for each initial region was arbitrarily chosen from the set of pixels in the region that were classified as red or gray with a value bright enough to exclude those pixels which could potentially belong to a shadow.

Once seed point pixels are assigned throughout an image, regions are "grown" from these seeds. Each seed pixel is compared to its adjacent pixels. If the adjacent pixel value is within a given threshold of the seed value, then that pixel is assigned to the corresponding seed. This is an iterative operation. Each seed region spreads as it continues to evaluate pixels neighboring its new members and compare them to the new mean value of that seed region. All seed regions are grown simultaneously. If a pixel has

already been assigned to another seed, it is not assigned twice. The iterative growing operations end when all pixels in the image have been uniquely assigned to a seed.

This method is expensive both computationally and in terms of memory. It is also difficult to implement in MATLAB®, as simultaneous operations are not possible. Another limitation to this method is that the number of resulting regions is not adaptive. The number of seed points is chosen up front, and the number of resulting regions is equal to the number of seed points regardless of relevance. If too few seed points are chosen, buildings could potentially be combined, which defeats the purpose of the segmentation step. Since post-processing techniques rely on successful and complete segmentation, combined buildings will be treated as one building, and the resulting extraction will have errors. If too many seed points are chosen, building extraction can still be successful with appropriate post-processing. However, as the number of seed points increases, the computational and memory costs increase as well.

2.2.3 Multithresholding and Clustering.

In Section 2.1 thresholding is discussed as a way of segmenting an image based on foreground and background assumptions and thereby converting the image to a binary image. This method is also known as bilevel thresholding [9]. Multithresholding, on the other hand, is necessary when the objects of interest vary in pixel values. In other words, the image cannot reasonably be divided into one foreground value and one background value. The simplest way to divide an image using multiple thresholds is to create bins of equal sizes within the 0 to 255 pixel value range and classify pixels based on the bins to which they belong. However, this method employs little intelligence about the values present in the image and can result in bins with far fewer members than other bins. It can also result in inadvertently grouping pixels which should be kept separate. One way of intelligently choosing appropriate threshold values for the image involves analyzing the histogram of the image for peaks and valleys and assigning threshold values based on

natural value groupings. However, image histograms rarely have definite peaks and valleys and if histogram equalization was applied during preprocessing to enhance contrast, the histogram will provide no useful information for choosing thresholds. Reference [9] does list many helpful tricks, however, which can be applied to an image histogram to emphasize groupings.

In addition to the lack of defined peaks, the lack of spatial information is another drawback of using only the histogram to determine thresholds. Therefore, [9] also lists many techniques that employ spatial information to accomplish thresholding. These techniques include something called the "busyness" measure and the "co-occurence matrix," which both provide information about the similarity of adjacent pixels. Entropy measures as well as the conditional probability of region transitions can also provide helpful information about similarity throughout an image. Combining this information can aid in choosing likely threshold values, which can then be used to extract objects. These methods are described with more detail in [9]. They incur varying computation and memory costs, but along with the histogram analysis techniques, they all share the same goal. They all attempt to make informed guesses about which threshold values will be most useful for multithresholding segmentation. Post-processing will still be required to classify regions of interest and to create clean and accurate object borders, so spending more on a sophisticated multithresholding technique may very well have diminishing returns.

All previous information in this subsection is extracted from [9], and all of the following information in this subsection is extracted from [4]. Another segmentation method similar to the concept of multithresholding is clustering. While thresholding determines cut-off values which separate one group from another, clustering determines a representative value for each group. K-means clustering is a common method of clustering which uses the mean value of each group as the representative value, and it is

called “K-means” because there are K clusters and therefore K means. With K-means come a few limitations. First, the number of clusters, K, must be chosen up front. In the case of objects in an overhead image, the user must make an educated guess about how varied the objects of interest in the image are. Guessing a high number leads to excess computation and memory costs, but guessing a low number may result in a combining of objects and a loss of data. The non-parametric nature of the K-means algorithm brings another limitation, which can also be an advantage. The algorithm does not rely on given assumptions about the distribution of pixel values in the image. This can lead to a less sophisticated result, but it also allows for a generalized method that does not require prior knowledge of the image.

K-means clustering is an iterative method which updates the mean of each group as pixels are re-allocated. Pixels are re-allocated with the goal of minimizing variance (σ_E^2), defined by

$$\sigma_E^2 = \sum_{k=1}^{K} \sum_{x \in X_k} \|x_m - \mu_k\|^2, \tag{2.3}$$

where μ_k is the mean of the kth cluster, and x_m is the value of a given pixel assigned to the current cluster. It follows then that M_k is the number of pixels assigned to cluster k. Pixels are reassigned until μ_k values converge, expressed as

$$|\mu_k(n+1) - \mu_k(n)| < \varepsilon \; \forall \; k = 1 \ldots K, \tag{2.4}$$

where n is the iteration number and ε is some small value determined by the user. K-means clustering can be implemented with a relatively simple algorithm, and MATLAB® already has a built-in K-means function. It can incur large computation and memory costs depending on the value of K, but it is one of the simpler methods of segmentation that also has adaptive abilities.

2.3 Image Post-Processing Methods

After segmenting an image into regions, further analysis is required to determine which regions are of interest. In this paper, interesting regions are groups of pixels which define a building. This section covers various current methods of determining potential buildings. Some methods are only applicable to regions defined by a certain image segmentation technique, but these relationships will be clear.

2.3.1 Spectral Patterns.

The first post-processing technique utilizes the frequency domain and is most useful when applied following edge detection segmentation. The information in this subsection is gathered from [11]. This source utilizes edge detection but then sorts through the edges based on an assumption about human tendencies toward organization. It posits that dense urban areas are constructed with "coherent" directionality, which can be seen in the fast Fourier transform (FFT) power spectrum of the image. This means that neighborhoods of buildings generally have a similar orientation, and the frequency domain can be used to find these dominant orientation angles from an overhead image. Figure 2.6 shows how the frequency domain can highlight patterns within an image. The bright lines through the spectral image describe the directions of greatest change in the spatial image. In other words, the greatest number of edges would be found traveling in these directions through the image. Most edges in a dense urban scene belong to sides of buildings. Therefore, the directions of greatest change in an image relate strongly to the axes of the buildings in the image.

Under the above assumptions of coherent directionality, Figure 2.6 illustrates how the frequency domain can be used to determine dominant orientation angles of buildings in an urban scene. These dominant angles are then used to discard extraneous edges with orientations that do not match the building orientation assumptions. They can also be used as a guide to adjust detected edges that may be slightly off the dominant angle or to

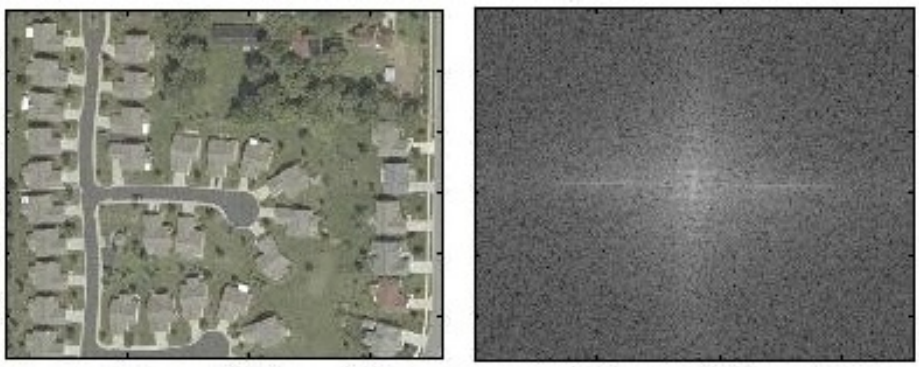

Figure 2.6: An example of a suburban area in (a) the spatial domain and (b) the compressed FFT power spectrum of the image created in MATLAB®. In an urban scene obeying the coherency assumption and containing a more dense population of building structures, the lines through (b) would be more defined than they are for this suburban example. Spatial data available from the U.S. Geological Survey [1].

lengthen and connect other edges which may be missing relevant pieces. Work in [11] completes post-processing by combining objects with the same pixel values under the assumption that building roofs are homogeneous in color. Using spectral patterns in this way can produce clean and accurate building boundaries. However, it disregards non-rectangular buildings as well as those buildings which may not follow the patterns it assumes. These patterns are more reliable in urban scenes with very densely packed buildings. Therefore, it would not necessarily make sense to apply this technique to all overhead images.

2.3.2 Defining and Merging Regions.

As the result of any method of segmentation, an image is divided into regions. Often these regions do not have smooth or clear border lines. Especially if thresholding or clustering is used for segmentation, there can be pixels belonging to one group of values peppered throughout a region mostly containing pixels from another group. Also,

segmentation often results in blob-like regions rather than well-defined geometric shapes such as buildings. Edge detection segmentation methods will likely avoid this issue, but applying a contour-defining algorithm may still improve results. Opening and closing operators can provide the desired definition to the regions. Opening involves first erosion and then dilation, while closing is the reverse process. Dilation and erosion are morphological operations which use a structuring element to comb through an image and either fill in gaps and expand regions or remove lonely clumps and contract regions, respectively. Reference [8] uses these morphological operations as well as the following merging method.

The merging method discussed here is developed by and described in [8]. Image segmentation often results in multiple regions that actually belong to the same object. Researchers in [8] found a way to determine which regions define parts of the same building. First they assume that all three-dimensional objects, including buildings, cast a shadow. They also assume that buildings are rectangular, and they therefore discard shadows without a straght edge. Then they dilate the segmented regions and determine if a particular shadow significantly overlaps more than one region. If the overlapping regions also overlap each other by an amount determined to be significant, then these regions are combined, and the new building region is eroded back to a normal size. One drawback to this merging method is that it fails to merge roof pieces of buildings that do not have a fully extended L-shaped shadow. For example, if the sun directly faces one side of a building then only the opposite side will have a shadow alongside it. Figure 2.7 illustrates how this situation would result in half of the roof having no contact with a shadow and therefore being missed.

2.3.3 Feature Extraction.

Unless otherwise specified, the information in this subsection is pulled from [8]. After defining and merging regions, [8] develops a list of features with numerical values to

Figure 2.7: An example of a situation when a shadow only extends along one side of a two-toned building. Data available from the U.S. Geological Survey [1].

quantitatively compare segmented objects in order to decide which of them are buildings. Since keeping an exhaustive list of feature values for each and every object can be time and memory consuming, [8] uses a couple "preselection" criteria in order to weed out improbable objects. Reference [8] assumes a minimum building area and a red roof hue and therefore discards objects that are too small or that have an average pixel value that describes a green hue. Once potential buildings have been preselected, [8] creates a list of over a hundred feature values for the remaining objects.

Some of the features used are geometric form features, including *roundness*, *compactness*, *lengthness*, and *characteristicangles*. Reference [8] defines roundness as a value between 0 and 1 calculated by

$$roundness = \frac{4\pi x area}{circumference^2}. \tag{2.5}$$

Roundness will be equal to 1 for a perfect circle and 0 for a line. *Compactness* relates to the shape, density, and thickness of an object and is equal to the number of times erosion would have to be applied to the object in order to erode it completely. Reference

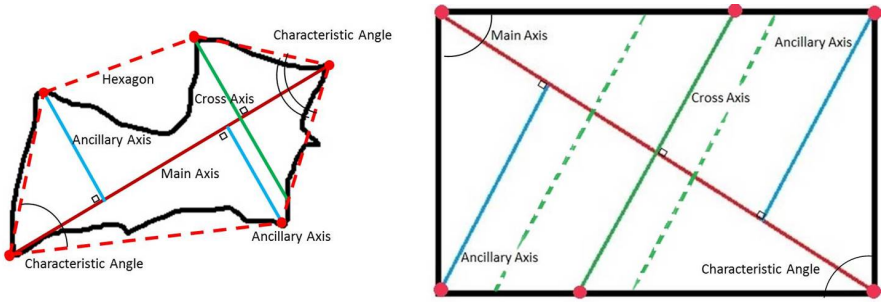

Figure 2.8: Illustration of the axes locations in (a) an organic object and (b) a rectangular object. For a rectangular object, the cross axis is irrelevant since there are infinitely many lines of the same maximum length that connect two border points and are perpendicular to the main axis.

[8] defines *lengthness* as the length of the main axis divided by the length of the cross axis. The main axis is found by drawing a line between the two border points that are farthest from each other. The cross axis is found by connecting the two border points farthest from each other which lie on a line perpendicular to the main axis. Two ancillary axes also exist parallel to the cross axis and on opposing sides of the main axis from each other. They each lie along the line reaching from the main axis to the farthest possible border point on their respective sides. The six border points used to create these axes form the corners of a hexagon. Reference [8] refers to the two corner angles of the hexagon located at the ends of the main axis as *characteristicangles*. If the object is rectangular as illustrated in Figure 2.8, the hexagon resulting from connecting the axes border points will actually describe the same rectangle that is created by the border of the object. Therefore the two *characteristicangles* will be right angles. Since [8] assumes most buildings are rectangular, they expect the *characteristicangles* of a building object to be nearly right. They also use the hexagon created by the axes points to develop a few other feature values useful in determining whether or not an object is likely to represent a building.

Other useful features can be photometric or structural. For example, [8] assumes that buildings are constructed in groups, and therefore isolated buildings are unlikely. Then as another feature, numerical values are assigned to describe an object's proximity to other likely building objects. It is also assumed that a shadow will be present next to a building, since quality aerial images can only be taken on a clear day. Shadows are extracted by assuming that pixels with a value below a certain threshold are dark enough to be considered shadows. This feature can take on a numerical value by counting the number of overlapping pixels between the object in question and a shadow object after a temporary double dilation has been applied. These are just a few examples of many interesting features that can be helpful.

The feature extraction method of decision-making provides thorough and reliable results as long as the conditions in the image follow the many assumptions made in the process. However, the assumptions do decrease the value of the method as a general building extraction algorithm. Furthermore, storing over a hundred values for every single potential building object may be impractical, especially when obtaining each value requires complex calculations. Making decisions to weed out objects as features are calculated may yield results with similar accuracy.

2.4 Geolocation Methods with Unobstructed Lines of Sight

The rest of this chapter concerns signal processing and specifically signal processing with the goal of geolocation. The most useful piece of signal information to use for geolocation is the signal time of arrival (TOA), but there exist many viable methods for utilizing this information to locate the source. This section focuses on different approaches for using TOA to locate a radio frequency (RF) signal source when there are no obstructions to the lines of sight between the source and the sensors.

2.4.1 Analytical Approach.

All of the information in this subsection is derived from [3] unless otherwise specified. Reference [3] discusses methods of using multiple sensors to locate a single source in a two-dimensional plane and discusses a way to use hyperbolic curves to accomplish this geolocation. The hyperbolic curves in question are those created by time difference of arrival (TDOA) estimates, defined as the relative differences in signal detection timing among the sensors. The relative time at which a sensor first detects a signal of interest is determined through cross-correlation, which essentially creates a copy of the signal and compares it to data received by the sensors [3, 10]. The detection time is defined as the location in time where the signal copy is aligned to match a section of data received by the sensor. Estimating TDOA is easier than directly estimating TOA since there is no way to know when the received signal was actually transmitted. A method is described which is accurate for both close sources and distant sources. It also utilizes an increase in the number of sensors to increase the accuracy. Some of the previous methods can only utilize TDOA estimates from a number of sensors equal to one greater than the number of unknown coordinates in the source location. In the two-dimensional case, this means those methods can only benefit from the information from two TDOA estimates and therefore three sensors. Reference [3] begins with an arbitrary configuration of M sensors and a single source with an unknown location in a two-dimensional plane. In order to find the location of the source, TDOA estimates are first generated using a cross-correlation technique and then estimated with respect to one receiver, which is thereafter referred to as the first receiver. The TDOA of the ith sensor is given by

$$d_{i,j} = d_i - d_1 \text{ for } i = 2, 3, \cdots, M, \tag{2.6}$$

which is used to form the estimated TDOA vector $\vec{d} = [d_{2,1}, d_{3,1}, \cdots, d_{M,1}]^T$. The covariance matrix of the TDOA vector is given by

$$\mathbf{Q} = \left\{ \frac{2T}{2\pi} \int_0^{\Omega} w^2 \frac{S(\omega)^2}{1 + S(\omega) tr(\mathbf{N}(\omega)^{-1})} x \left[tr(\mathbf{N}(\omega)^{-1}) \mathbf{N}_p(\omega)^{-1} - \mathbf{N}_p(\omega)^{-1} \mathbf{1}\mathbf{1}^T \mathbf{N}_p(\omega)^{-1} \right] d\omega \right\}^{-1}, \tag{2.7}$$

where 0 to Ω is the frequency band, T is the observation time, $tr(*)$ denotes the trace of matrix $*$, $S(w)$ is the signal power spectrum, $\mathbf{N}(\omega) = diag\{N_1(\omega), N_2(\omega), \cdots, N_M(\omega)\}$ is the noise power spectral matrix, $\mathbf{N}_p(\omega)$ is the lower right $M-1$ by $M-1$ partition of matrix $\mathbf{N}(\omega)$, and $\mathbf{1}$ is a unity vector with length $M-1$. The noise has a mean of zero and a covariance matrix equal to $\mathbf{Q}$. The unknown position of the source is denoted (x, y), the known location of any sensor i is denoted (x_i, y_i), and the Euclidean distance between the source and sensor i is denoted r_i. Therefore,

$$r_{i,1} = cd_{i,1} = r_i - r_1, \tag{2.8}$$

where c is the signal propagation speed. Since the only unknown values are the source x and y, which are within the expression for r_i, the solution to this system of equations yields the location of the source. Equation (2.8) looks simple, but the solution is actually very complex, because the system of equations is nonlinear. Two of the possible approaches to solving the system are linearization through Taylor series expansion and spherical interpolation (SI). The Taylor series expansion method is more accurate than SI, but requires solving iteratively after linearization which can be costly. To linearize by Taylor series expansion, an initial source position (x_0, y_0) is guessed, and then after each iteration the position guess is updated by a deviation amount $(\Delta x, \Delta y)$ calculated within the iteration by

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = (\mathbf{G}_t^T \mathbf{Q}^{-1} \mathbf{G}_t)^{-1} \mathbf{G}_t^T \mathbf{Q}^{-1} \vec{h}_t, \tag{2.9}$$

where

$$\vec{h}_t = \begin{bmatrix} r_{2,1} - (r_2 - r_1) \\ r_{3,1} - (r_3 - r_1) \\ \vdots \\ r_{M,1} - (r_M - r_1) \end{bmatrix},$$

$$\mathbf{G}_t = \begin{bmatrix} \frac{x_1 - x}{r_1} - \frac{x_2 - x}{r_2} & \frac{y_1 - y}{r_1} - \frac{y_2 - y}{r_2} \\ \frac{x_1 - x}{r_1} - \frac{x_3 - x}{r_3} & \frac{y_1 - y}{r_1} - \frac{y_3 - y}{r_3} \\ \vdots & \\ \frac{x_1 - x}{r_1} - \frac{x_M - x}{r_M} & \frac{y_1 - y}{r_1} - \frac{y_M - y}{r_M} \end{bmatrix},$$

where $\mathbf{Q}$ is defined by Equation (2.7), and $x = x_0$, $y = y_0$. Iteration continues until the deviations are smaller than a pre-determined threshold, and the final (x, y) values are the solution for the source location.

SI deviates from Taylor series expansion by squaring Equation (2.8) and expanding r_i^2 in terms of x, y, x_i, and y_i. Then, $(x_1^2 + y_1^2)$ is subtracted from both sides of the equation, and x and y are solved in terms of r_1. The intermediate result is inserted back into the same equation that was just used to solve for x and y so that r_1 is the only unknown. A value of r_1 is found to minimize the least-squares (LS) equation error, and this value is substituted into the intermediate result for x and y to find the source location. The SI method does not require iteration, but it does require LS calculations which are not trivial, and according to [3] the solution is not optimum.

2.4.2 Numerical Approach.

Another method for estimating the location of a signal source using TDOA is a maximum likelihood estimator (MLE) technique known simply as a grid search. This method is decribed in [5] and is as simple as it sounds. This method is similar to a "guess-and-check" method, except that it systematically guesses every possible answer if the possibilities are discrete and finite. If the possibilities are continuous values or are an

infinite set, it is impossible to guess every potential answer, but small step sizes can be chosen to mitigate the error. The way the guess is checked is by finding the difference, $diff$, between the actual data and the data which would be produced if the guess were the true location. This difference is defined as

$$diff(\hat{\theta}) = E\left[(\hat{\theta} - \theta)^2\right], \tag{2.10}$$

where $\hat{\theta}$ is the estimated data and θ is the actual data. Then whichever guess produces the smallest $diff$ is the most likely solution. This method is especially applicable to image processing, since locations are defined in terms of discrete pixels, and there are a finite number of pixels. In reality, the pixels in an overhead image correspond to small regions of ground locations, which are continuous values. However, as long as the overhead image encompasses the locations for all sensors and for the source, it is actually possible to check every single pixel and therefore, under ideal conditions, to guarantee that the MLE for the location in the image is found within a measurement resolution equal to half of the ground sample distance. The downside to this method of course is that checking every single pixel is time consuming, and the computations involved in finding $diff$ at each pixel can be costly. The method does require very little storage though, since it does not need to store information about the guesses that do not produce the smallest $diff$.

2.5 Fusion of Signal Information with Image Information

This section focuses on the novel idea of how to incorporate the discoveries made through image processing in order to more accurately process the signal information when obstructions do exist in the line of sight (LOS) between the source and the sensors. If the signal cannot travel in a straight line from the source to the sensor, then it must take another route to the sensor. This route will most likely involve a reflection off of another building in the scene, which will delay the signal TOA. Thanks to the image processing techniques discussed earlier in the chapter, the locations of the buildings in the scene can

be extracted. Therefore, obstructions as well as potential reflection points are known. The following subsection discusses a method of finding the shortest path from a known source location to a single sensor. This concept can then be extended to multiple sensors and to a source with an unknown location.

2.5.1 Shortest Path.

Information in this subsection is provided by a faculty member at Air Force Institute of Technology (AFIT) in the form of an unpublished draft specifically as a contribution to this research [7]. The goal is to find the shortest unobstructed path between a transmitter and a sensor, both with known locations. In this thesis, the transmitter location is not in fact known, but it is easier to understand the method of finding the shortest unobstructed path if the problem is first approached using known locations. Here path length is measured in Euclidean distance, since geolocation is dependent on TOA, and a signal's travel time is directly proportional to distance. The problem is approached in [7] with the assumption that there are multiple receivers and transmitters, all with known locations. However, this subsection focuses solely on the case of one receiver and one transmitter with known locations, since the extension is straightforward and discussed in part in the following subsection. Nomenclature refers to the obstructions as well as the reflective objects as buildings, and reflective surfaces are referred to as walls. Of course since the problem remains in a two-dimensional plane, walls are simply lines with finite lengths. This dimensionality is valid, since it is safe to assume that any building will be tall enough to intercept the signal and also that in an urban environment the ground is flat enough not to interfere with the signal.

The problem would be even simpler if buildings could be treated as points in the plane, but unfortunately that would be a gross oversimplification. An acceptable simplification that can be made, however, is the assumption that the RF signal emitted by the transmitter does not obey the law of reflection. To provide a memory refresh from high

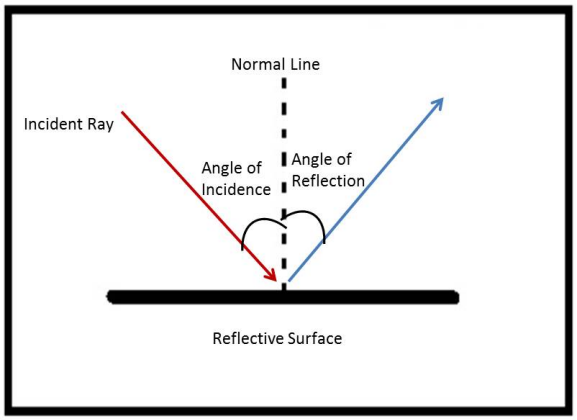

Figure 2.9: Illustration of the law of reflection.

school physics, the law of reflection states that the angle of incidence is equal to the angle of reflection. Figure 2.9 illustrates the locations of these angles. In the case of RF waves and building walls, however, [7] explains convincingly that Lambertian reflection occurs instead. Due to the polarization of RF waves and the roughness of building walls, the angles of incidence and of reflection are reasonably unpredictable. Fortunately, as the roughness of a surface increases, thereby increasing the complexity of the problem, the more the surface actually scatters the RF waves, which allows the problem to be assumed away. The rougher a surface is, the less concentrated the reflection is in a specific direction. The reflected signal is therefore assumed to be reflected with equal strength in all outward directions.

When searching for the shortest path, the first step is to check for obstructions. The test for obstructions involves checking if any wall lines intercept the line between the receiver and the transmitter. If the receiver has an unobstructed LOS to the transmitter, then the shortest path is along that LOS, and no extra calculations are required in order to find the path. However, if the path is obstructed, all alternate paths must be considered and compared to one another.

To find the alternate paths, the first step is to determine which walls are "visible" to the transmitter. "Visible" walls are walls that are positioned and oriented in a way such that the transmitted signal can be reflected off the wall. In [7], visibility is determined using the normal line of each wall, like the one pictured in Figure 2.9. The unit normal vector pointing out from a particular wall is denoted $\hat{\Omega}_w$, where w denotes the particular wall. Other data specific to each particular wall are the end points and the midpoint of that wall. In [7], $\hat{\Omega}_w$ is found by using the wall endpoints in conjuction with the property of cross-products which says that the magnitude of the cross-product of two perpendicular vectors is equal to 1. This property is helpful, since the wall endpoints can be used to create a unit vector which describes the direction along the wall and therefore perpendicular to the normal vector. The unit vector along the wall must point from the left end of the wall to the right end, as defined when facing the wall from the outside of the building. Keeping track of left and right in this manner is significant, because flipping this vector will result in a normal vector pointing into the building, which will lead to inaccurate reflection results.

The information describing each wall is then stored in an exhaustive list containing all walls. Keeping track of which walls belong to the same buildings is not necessary since the outward direction and location of the wall is all that is needed to determine reflections. Once $\hat{\Omega}_w$ is calculated for each wall, it can be used to determine visibility. To determine if a wall with normal vector $\hat{\Omega}_w$ and midpoint $\vec{r}_w$ is visible to both an emitter point $\vec{a}$ and a receiver point $\vec{c}$, [7] uses the following two conditions, which are also illustrated in the Figure 2.10 example.

$$(\vec{a} - \vec{r}_w) \cdot \hat{\Omega}_w > 0 \text{ and } (\vec{c} - \vec{r}_w) \cdot \hat{\Omega}_w > 0 \tag{2.11}$$

It is important to remember that an emitter point $\vec{a}$ in this case can refer to either the original transmitter or to another point from which the signal has been reflected. Likewise,

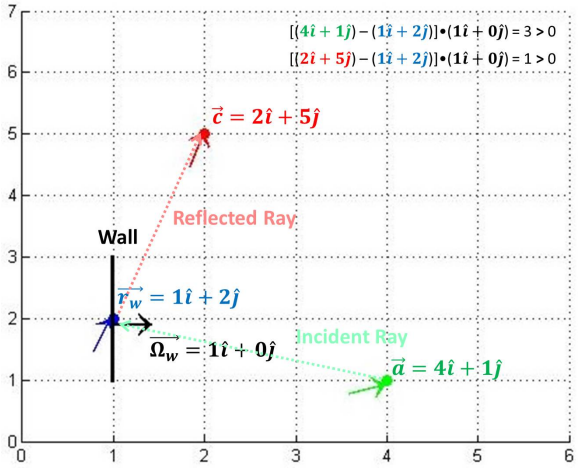

Figure 2.10: Example illustration of Equation (2.11).

$\vec{c}$ denotes either a receiver or a point to which the signal will travel for another reflection. Also, the midpoint of the wall $\vec{r}_w$ is specified in Equation (2.11) for ease of reference and simplicity. In reality, any point on the wall can be used as the reflection point, since it is assumed that the signal is emitted in all directions from $\vec{a}$. The optimum reflection point on the wall can be calculated instead of using the midpoint, but the impact on the path length is not significant. Equation (2.11) can be applied at each leg of each path to find all possible paths from the transmitter to the receiver. The complexity comes from two issues. The first is that each leg of each path must be checked for obstructions. The second issue pops up if one of the path legs is found to be obstructed. Just because one point on a wall is obstructed from a point on another wall does not mean that all points on both walls are obstructed from one another. Therefore, if an obstruction is found, all possible reflection points must also be checked for obstructions before disregarding that path.

There are a few options for keeping track of paths and their lengths. A distance matrix can keep track of each unobstructed view between walls and other points. Keeping track this way cuts down on computation time since path leg calculations are only calculated once and can be reused for alternate paths. This can be especially useful when the

problem is expanded to incorporate multiple sensors. However, a distance matrix like this also takes up a lot of memory. To reduce storage, [7] also suggests using dynamic storage in the form of linked-lists. This way, the distance between pairs of points is only calculated if it will be used, and then it is stored to eliminate recalculation. Another option is to store only the shortest path length and compare each new path calculation to the stored value. For geolocation using signal TDOA, only the time it takes for the signal to travel the path has significance. Which path the signal takes does not matter at all. This cuts down the storage significantly but also increases computation time since path legs used in multiple routes are calculated multiple times.

Another way to cut down computation cost is to actually keep track of which walls belong to which buildings. Since no more than two walls from the same rectangular building can be visible from any single point, eliminating the need to check every wall reduces computation time. Approaching the problem this way, however, does have further implications. Other modifications would have to be made to the process to make this method work. Reference [7] outlines a method which involves treating buildings as nodes while taking into consideration that unlike a node, a building takes up space. This method requires knowledge about the shape of every building, and does add complexity in that respect. The method used later in this paper avoids this added complexity as well as the excess storage involved in a distance matrix, but unfortunately it does involve calculating some distances multiple times.

2.5.2 Applying Multipath Analysis to Geolocation.

Extending the current research and concepts discussed in this chapter and combining them to improve upon previous methods of geolocation is a novel exploration and is the goal of this thesis.

III. Methodology

This chapter discusses the process used to develop a transmitter geolocation tool which incorporates both image information and signal information. The chapter begins with a description of building extraction from overhead, aerial, orthorectified RGB images. The algorithm as a whole is an original creation, but the parts which make up the algorithm are not original. Each step and technique has been utilized in other research in some way. This building extraction algorithm simply seeks to find an effective and efficient combination of techniques in order to extract image information which can be used to improve geolocation. The only inputs required for the algorithm are the particular image, an estimate in pixels of the smallest expected building perimeter in the image, and an estimate in pixels of the largest expected building perimeter in the image. These two estimates are not essential and in fact should be made conservatively, but including them significantly decreases memory and computation costs as well as the possibility of returning false positives. Section 3.1 presents the preprocessing tools employed to efficiently and effectively utilize image information. Section 3.2 explains the type of image segmentation used, including how shadow objects are extracted and processed. Then, Section 3.3 discusses how information about the image segments is utilized in conjunction with information about shadow objects to determine the locations of buildings in an image.

The remainder of the chapter discusses how the information about building locations is used to improve transmitter geolocation. Section 3.4 discusses how these locations are used to determine which sensors have an obstructed LOS to the transmitter. Section 3.5 discusses how to find the shortest path between a transmitter and a sensor, even if that path includes reflections from building walls. Section 3.6 concludes the chapter with a description of how paths which include reflections can be incorporated into the TDOA

Figure 3.1: Raw image data which has been orthorectified. Data available from the U.S. Geological Survey [1].

estimation process to improve performance. The techniques described to perform traditional geolocation as well as those used to find path information are not original. However, the combination of these techniques with the image information is an original pursuit.

3.1 Image Preprocessing

Throughout the following discussion of image processing for building extraction, each step is applied to the image in Figure 3.1 and illustrated in corresponding figures.

The first step in working with this image requires converting the image to a grayscale intensity image. This allows efficient manipulation of data and ease of comparison among pixel values. The RGB values in Figure 3.1 are converted to V channel intensity values in the HSV color model. This conversion is accmplished by averaging the three channel contents from the RGB color model. The result is shown in Figure 3.2. Also shown in Figure 3.2 is the result of applying contrast enhancement to the intensity image. Before enhancing the contrast in the image, a simple median filter with a 3×3 kernel is applied to

Figure 3.2: (a) The intensity channel of the HSV color model and (b) the image after contrast enhancement. Original image data available from the U.S. Geological Survey [1].

lightly add smoothing while retaining edge integrity. The method of contrast enhancement used involves mapping intensity values to new values in order to stretch the present range of intensity values onto the full available range with 0 as the lowest value and 255 as the highest value. This is accomplished by saturating the lowest 1% of the intensity values at 0 and the highest 1% of the intensity values at 255.

3.2 Image Segmentation

This building extraction algorithm relies on the assumption that every building casts a shadow. Therefore, the image segmentation process first creates a set of pixels likely to belong to shadow objects and then clusters the remaining pixels. All pixels below an intensity threshold value of 50 are considered dark enough to have the potential to be shadows. These pixels are separated from the original image, and a new black and white (BW) image of the same size is created with the shadow pixels as logic 1's and all remaining pixels as logic 0's. This BW image is then morphologically "filled" to remove holes and "closed" to connect pixels which may belong to the same shadow. Then, all the pixels in the original image which correspond to the logic 1 pixels in the BW image are

set to zero, as a way of separating these potential shadow pixels from the remaining pixels and also as a form of labelling. The following subsection describes the clustering process for the non-zero, non-shadow pixels, and the subsequent subsection describes how the set of shadow pixels is further processed.

3.2.1 K-means Clustering.

All remaining non-zero pixels in the image have the potential of belonging to a building object. Since building roofs tend to be largely monochromatic, grouping similar pixel values is an effective way to segment an image into potential building objects. This grouping is accomplished using K-means clustering, which is described in full mathematical detail in Section 2.2 of *Chapter II*. A K value of 6 is used in this algorithm, since an even value is ideal for subplotting purposes. Clustering the pixels into 4 sets proves inadequate for creating the necessary separation between objects, and clustering into 8 sets requires an excessive amount of computing power and memory. Also, increasing the number of cluster sets breaks building objects into smaller and smaller pieces, which makes them difficult to completely recover later on.

MATLAB® contains a built-in function which accomplishes K-means clustering. Inputs to the function are used to instruct the function how to proceed with exception cases and how to choose the initial K means. If throughout the iterative clustering process, a cluster set loses all its members, the pixel in the image which has a value furthest from the mean pixel value of the whole image is placed into this set in order to avoid empty sets. The initial K means are chosen randomly using a uniform distribution with a range equal to the range of pixel values being clustered. Uniformly choosing the initial means helps ensure separation among dissimilarly colored objects. On top of these exception and initialization inputs, another input is included in the function which replicates the clustering three times, each with different initial means, and chooses the clustering solution which results in the smallest error of the three. The error in each solution is

Figure 3.3: Image after clustering has been accomplished. Original image data available from the U.S. Geological Survey [1].

calculated by summing the distances between each pixel and its corresponding cluster mean. Since the clustering process contains random elements, the potential exists for pixels to be allocated differently each time. This replication step limits the variation in the overall building extraction algorithm results. The downside, however, is that K-means clustering is expensive to begin with, and replicating in this way may not be possible. The processor used in this research is capable of replicating a maximum of three times due to the length of the variables, and in cases of larger images, replicating is not possible at all. However, with a more powerful processor, replicating as many times as possible is recommended to produce more consistent results.

Since the pixels which have been set to zero have already been allocated to the shadow set, they are ignored during the clustering process. Figure 3.3 shows the image after clustering is accomplished. Each pixel value is set to equal the mean value of the cluster set to which it belongs, and the pixels in the shadow set are set to zero.

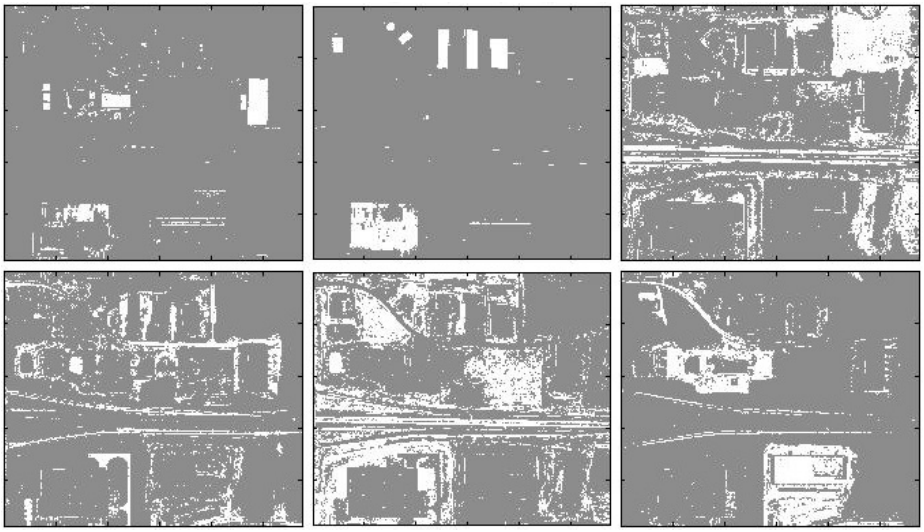

Figure 3.4: Individual sets of clustered pixels. Original image data available from the U.S. Geological Survey [1].

After clustering is accomplished, a new BW image of the same size as the original image is created for each cluster set. Within each BW image, the pixels belonging to that cluster set are set to logic 1's. Figure 3.4 shows how these cluster sets have been separated.

Once the cluster sets are separated in this way, they can be looked at individually. Each BW image is essentially morphologically "opened" in order to increase separation among objects. Especially in the case when a building is the same color as the surrounding area in the image and the edges are thin, the edge pixels may not all be detected. This can cause building pixels to connect undesirably with surrounding pixels. It is essential to remove as many of these undesirable connections as possible without damaging each object. Since a morphological "opening" is basically an erosion followed by a dilation, this step is expanded to increase effectiveness. A double erosion is followed by a removal of "H" connections and spurs, as well as by a "clean" operation, by an actual "open" operation, and finally by a double dilation to return the objects to their original size. Both erosion and dilation are accomplished using a 3×3 structuring element. The

"clean" operation simply removes any isolated pixels which are not connected to any other pixels. "Spurs" are pixels which are isolated aside from a single connection point, and "H-connected" pixels are those which form the connection between two lines of pixels. The following is an example of how H-connected pixels are removed:

$$\begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{matrix} \quad \text{becomes} \quad \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}. \tag{3.1}$$

Once these morphological operations are applied, individual objects are found by tracing the exterior boundaries of connected pixels within each BW image. Once all the objects have been segmented, the cluster set to which they belong is no longer relevant information. Finally, each object is compared to the provided estimates of the minimum expected building perimeter and of the maximum expected building perimeter. Since this is still an early stage of the algorithm, potential building objects should be discarded very conservatively. Therefore, only objects with perimeters less than $\frac{1}{8}$ of the estimated minimum or greater than 4 times the estimated maximum are discarded.

It is especially important to be conservative with the maximum perimeter allowed, since object edges can be jagged, which causes the perimeter calculation to be deceptively large. Using the perimeter of the convex hull of the object rather than the perimeter of the object itself would mitigate the inflation issue caused by the jagged edges. However, finding the convex hull of the object is more time-consuming and computation-intensive than simply using the perimeter of the object that is already found during the boundary-tracing step. Perimeter comparisons are made multiple times throughout the algorithm, and the issues presented by irregular shapes or jagged edges are not prevalent enough to warrant the added complexity of checking the convex hull each time. Also, due to the number of clusters, the image is segmented into so many pieces that it is rare for an object to be larger than a building. Objects that are too large usually represent a forest, a

long road, or some sort of border around the image. Even a conservative maximum threshold will discard these objects. As long as vital building pieces are not discarded, the exact thresholds used in this step are not important. While this step does mitigate the possibility of returning false positives later on, the purpose of this step is mainly to decrease memory and computation costs by decreasing the number of objects which must be considered in Section 3.3.

3.2.2 Shadow Segmentation and Processing.

The building extraction algorithm places a lot of emphasis on shadows. This subsection discusses how the set of pixels which have been set to zero are further processed into shadow objects. Initial shadow objects are found by tracing exterior boundaries of the connected shadow pixels. Then shadow objects that are too small or too large are discarded. Just as with the initial discards of building objects in the previous subsection, these discards are mainly to reduce processing time and memory cost. Therefore, these discards are conservative as well. Any object with a perimeter smaller than $\frac{1}{4}$ of the smallest expected building perimeter or larger than twice the largest expected building perimeter is discarded. The smaller value is based on the assumption that a shadow will overlap at least one side of the building. Therefore the length will be at least $\frac{1}{4}$ of the building perimeter. Since the perimeter of the shadow will be at least double its length, this minimum threshold is conservative enough to avoid discarding relevant shadow objects. The same conservative buffer exists for the maximum threshold used, since a shadow shouldn't overlap more than two sides of a building. This preliminary weeding process decreases the computation time for the remaining shadow processing steps. The pixels which still have the potential of belonging to shadow objects are highlighted in Figure 3.5.

It is clear from Figure 3.5 that at this point, there still exist many pixels in the shadow set that do not actually belong to shadow objects. However, before the objects are

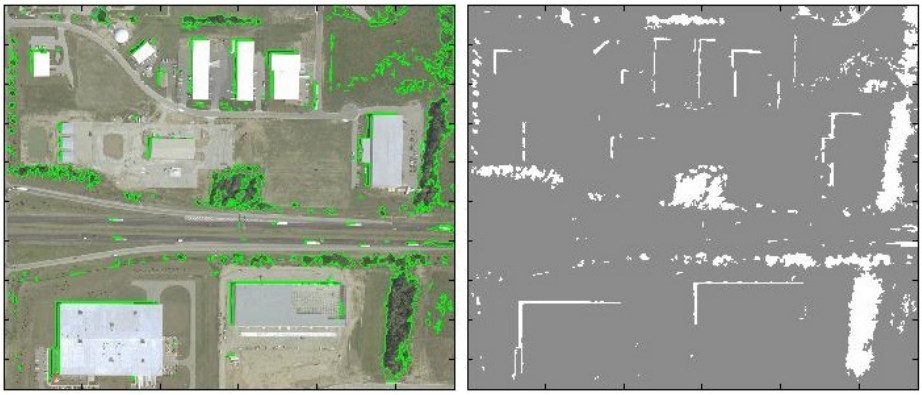

Figure 3.5: Pixels which have the potential to belong to shadow objects. Original image data available from the U.S. Geological Survey [1].

processed for shadow characteristics, it is important to combine any pieces that may belong to the same shadow, since the defining shadow characteristics include object descriptors such as shape and area. In comparison to Figure 3.4, Figure 3.5 shows that there are much fewer pixels in the shadow set than in any of the object sets. Due to this scarcity, there is less of a danger of connecting pixels which should not be connected. Therefore, a double dilation is applied, followed by a closing operation.

Individual shadow objects are then found by tracing the exterior boundaries of connected pixel regions. A few preliminary conditions are then applied to each of these objects to discard those which are unlikely to represent shadows. The conditions required for retaining an object include an eccentricity greater than or equal to 0.8, an area less than or equal to $\frac{1}{4}$ of the maximum expected building area, and as before, a perimeter between $\frac{1}{4}$ of the minimum expected building perimeter and double the maximum expected building perimeter. The eccentricity of an object is equal to the eccentricity of the ellipse which has the same second-moments as the object. The eccentricity of this ellipse is equal to the ratio of the distance between the foci of the ellipse and the length of the major axis of the ellipse. The eccentricity of a circle is zero, and the eccentricity of a line is one.

Since a shadow lies along the sides of a building, it should be shaped more similarly to a line than to a blob. Even if a shadow object lies along two sides of the building and is more L-shaped, it will still have an eccentricity very close to one. The area condition is based on the assumption that a shadow is much smaller than the corresponding building. The maximum building area estimation to which the shadow area is compared is found simply by squaring $\frac{1}{4}$ of the maximum expected building perimeter. Once again each of these conditions in this preliminary discard step is fairly conservative, but when the conditions are applied together, they are discriminating enough to remove a large portion of unlikely shadow objects. This is useful for the following processing steps, since computation time increases significantly with the number of shadow objects which must be processed.

Now that all of the remaining objects have a strong likelihood of representing shadows, it is helpful to determine which side of each shadow is the side which overlaps the respective building. Later on, when the algorithm searches for building objects which overlap the shadow objects, the algorithm needs to know which direction to travel in order to look for this overlap. This direction is referred to as *shadowdirection*. It is safe to assume that *shadowdirection* will be the same for all shadows and buildings within the image. This is not technically true, since the Earch is round, and Sun rays can be modeled as emitting from a point source, due to the extremely large distance between the Sun and the Earth. However, also due to the extremely large distance between the Sun and the Earth, as well as to the small scale of the images to which this building extraction algorithm applies, the differences in ray angles are negligible.

It is therefore possible to find the *shadowdirection* by averaging the directions calculated from each shadow and building pair throughout the image. This direction is essentially the angle of a vector which can be thought of as the vector pointing from the corner of an L-shaped shadow to the center of the building rectangle. Unfortunately, this vector is not as simple to find as it sounds, especially since not all shadows are L-shaped

depending on the orientation of the buildings, and also because the locations of the building centroids are unknown at this point. Therefore, this step relies solely on information about each potential shadow object. Fortunately, the manner in which *shadowdirection* is used later on does not require extreme precision. The algorithm essentially just needs to know whether to look up, down, left, or right.

The angle for each shadow object is found by taking the angle of the vector which points from the centroid of the shadow object to the midpoint between the two adjacent vertices furthest from each other in the convex hull of the shadow object. The convex hull is the smallest convex polygon which fully encloses the object. For a non-jagged, perfectly L-shaped object with singular thickness, this hull will describe a triangle. If this perfectly L-shaped shadow object completely overlaps two sides of the corresponding building, the hypotenuse of this hull triangle will cut straight through the building, with the midpoint of the hypotenuse in the same location as the centroid of the building. Even if the shadow object does have thickness, jagged edges, and is only vaguely L-shaped, the two adjacent convex hull vertices which are furthest from each other will still be the endpoints of a line segment which essentially describes the hypotenuse of the ideal case. Ideally, the vector of interest should point from the corner point of the "L" to the midpoint of these vertices, but this corner point is very difficult to find in a jagged and only vaguely L-shaped object with thickness. This is why the centroid of the object is used instead. It is very simple to find and provides separation from the endpoint of the vector. The point being used for the endpoint of the vector lies on the exterior boundary of the convex hull and on the side of the shadow object which overlaps the building. The centroid of the shadow object is within the convex hull. Therefore, a vector pointing from the centroid of the shadow object to the point found on the convex hull of the shadow object will point in the same general direction as a vector pointing from a shadow to its corresponding building.

What if all the buildings in the image are oriented in such a way that there are no L-shaped shadows? Even if all of the shadows only overlap one side of their respective buildings, the above method of finding the vector angle of interest is still valid. Intuitively, the furthest adjacent vertices of the convex hull of a rectangular shadow object will be the points describing the two longest sides of the rectangle, which in the case of the shadow will run parallel to the overlapping building side. The fact that the shadow objects likely do not describe perfect rectangles is actually helpful. The side of the shadow object which overlaps the building is much more likely to be a straight edge than the opposite side. The more jagged opposite side will create extra vertices in that side of the convex hull. Therefore, the longest side of the convex hull of the shadow object should still be on the side which overlaps the building. Once again, the centroid of the object will be within the convex hull, which allows enough separation to create a vector from the centroid of the shadow object to the midpoint of the longest side of the convex hull of the shadow object. This vector will point in the same general direction as a vector pointing from a shadow to its corresponding building.

The direction of this vector is important for finding pieces of the buildings later on in the algorithm, but it can also be used to help further process the shadow objects. Unfortunately, this is somewhat of a catch-22. At this point, there potentially still exist objects in the shadow set which are not actually shadows. Using *shadowdirection* to weed out these objects seems paradoxical when these objects played a role in finding *shadowdirection* to begin with. In order to filter out this corruption, *shadowdirection* is therefore recalculated. Remember, *shadowdirection* is the average direction of all the directions found from the individual objects. It is safe to assume that the majority of the shadow objects are actually shadows. As discussed above, a precise *shadowdirection* is not a necessity, but precision can still be improved with an iteration step. The value of

shadowdirection is therefore recalculated ignoring any of the directions which generally point in the opposite direction from the previously calculated *shadowdirection*.

Now that *shadowdirection* is a little more precise, it can be used to discard objects which likely do not represent shadows. Once again, the longest side of the convex hull of each object is found, and it is compared to the other points in the object. For ease of computation, the midpoint of this side is compared to the other vertices of the convex hull. Using the midpoint as the origin reference, if any of the other vertices lie within the same quadrant as *shadowdirection*, then this object is discarded. The assumption here is that this quadrant should be empty of shadow pieces, since it is where the building is located. It is explained above that shadow objects which are more rectangularly shaped than L-shaped should still theoretically create a convex hull with its longest side located along the building. However, there is a larger potential for error with these shadows than with L-shaped shadows. Therefore, before discarding an object that does not meet the quadrant requirement, the object shape is evaluated. If the object not meeting the quadrant requirement has its centroid located within the object and also has an extent ratio greater than 0.7, then the object is retained. It is explained above that the centroid of an object will always be located within the convex hull of the object, but this does not mean that the centroid will necessarily be located within the object itself. This is, however, likely to be true only if the object is rectangularly shaped, due to the earlier eccentricity requirement. Unfortunately, the centroid location is not enough to claim that the object is a shadow. For example, a dark tree line may be rectangularly shaped with an included centroid. This is when the extent requirement comes into play. A tree line will be much more organically shaped than a building shadow, which will appear as a more filled-in rectangle. Therefore, the shadow will have a larger extent, which is the ratio of the pixel area of the object to the pixel area of the convex hull of the object.

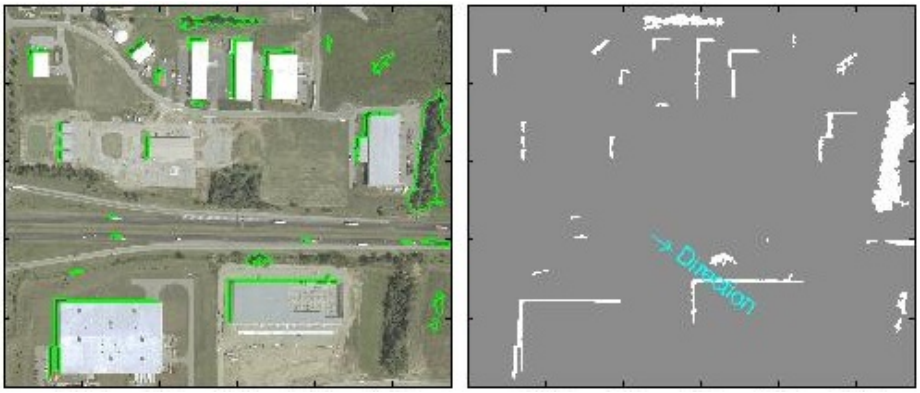

Figure 3.6: Final set of shadow objects. The direction labelled in cyan describes the direction of travel from each shadow to find the corresponding building. Original image data available from the U.S. Geological Survey [1].

The final set of shadows which are retained along with *shadowdirection* are shown in Figure 3.6. It is evident that there are still some objects retained in the shadow set that are not building shadows, but this does not necessarily mean that erroneous objects will be picked up as buildings later. There is still a lot of building processing which occurs in Section 3.3 to eliminate these.

3.3 Image Post-Processing

Now that the image has been segmented into shadow objects and possible building objects, and the direction from the shadows to their respective objects has been calculated, it is time to use all of this information to separate out the objects which most probably represent buildings or parts of buildings. The basic concept involves searching for those objects which overlap the shadow in the direction of *shadowdirection*.

Toward that end, the shadow objects are manipulated and eroded in two separate and different ways. In one way, the shadow object is eroded until only the edges which are likely to overlap the buildings remain. In the other way, the filled convex hull of the

Table 3.1: Shadow Erosion Structuring Elements

Unit Circle Section	Direction from Shadow to Object	Structuring Element
Up	$67.5° \leq shadowdirection \leq 112.5°$	$[0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1]^T$
Down	$-67.5° \geq shadowdirection \geq -112.5°$	$[1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]^T$
Left	$\lvert shadowdirection \rvert \geq 157.5°$	$[0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1]$
Right	$\lvert shadowdirection \rvert \leq 22.5°$	$[1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$

shadow object is eroded to a diagonal which is likely to cut through the building which corresponds to the shadow. Once these two manipulations are performed separately on the shadow object, a search is performed for building objects which overlap either of the remaining sets of pixels. This process will be explained further in the following subsections and illustrated in Figure 3.7, but before the shadow objects can be manipulated and eroded, first the structuring element(s) used for the erosion must be determined. The value of *shadowdirection* is used to determine which structuring element is used to erode the individual shadow objects. The unit circle is divided into eight uniform sections, referred to intuitively as "up," "down," "left," "right," "up left," "up right," "down left," and "down right." Table 3.1 shows the structuring element dictated by each of the four basic eighths of the unit circle. If *shadowdirection* lies in one of the remaining four compound eighths instead, then erosion is simply performed twice, once with each respective structuring element. This double erosion is not actually a compound erosion, however. Instead, each erosion is performed on the original shadow object, and the remaining pixels from each erosion are recombined. If the double erosion were applied compoundly, relevant shadow edges could potentially be lost.

3.3.1 Erosion of Shadow Object and Overlap Search.

Before the shadow object is eroded, it is dilated once with a basic 3×3 structuring element of 1's to cause a slight encroachment into the region where the shadow is likely to overlap building objects. Then the shadow object is eroded using the structuring element found above. This erosion is performed manually and differs from the traditional method of erosion. The structuring element is slid like a window across the entire object, and a new object is created in which the only pixels which are set to 1 are those which correspond to the center pixel of a set in the original object which "matches" the structuring element. A "match" here is defined as a set which contains at least one 1 with none of the 1's lying in the locations which correspond to the 0's in the structuring element.

Then, to further erode the shadow object towards its edges of interest, even more pixels are discarded. These discarded pixels are determined by a rule which is dictated by *shadowdirection*. This time, the unit circle is divided into quarters corresponding to the quadrants of the Cartesian coordinate system. Then, each remaining pixel in the shadow object is analyzed one at a time. Treating the pixel as the origin, if any other pixels in the shadow object lie in the same quadrant as *shadowdirection*, then the origin pixel is discarded. Ideally upon completion of this operation, the shadow object is reduced to sparse line segments which describe the edges of the shadow furthest in the direction of *shadowdirection* and therefore most likely to overlap the building which created the shadow. This step not only mitigates the risk of retaining erroneous pixels from an oddly shaped shadow object, but it also helps reduce the computation time later on by reducing the number pixels which must be checked for overlap with the building objects.

At this point, the shadow object is actually eroded even further. Due to the way this program searches for overlaps, which will soon be described in this subsection, there is a risk of picking up erroneous objects near the endpoints of the shadow skeleton. To

mitigate this risk, an eighth of the the remaining shadow pixels are discarded at each of the two extremes. These extreme points are found by once again using *shadowdirection*. The pixels are first analyzed with regards to the horizontal coordinate and then again with regards to the vertical coordinate. If the cosine of *shadowdirection* is positive, then the pixels discarded are those located at a horizontal coordinate within the maximum eighth of the horizontal coordinates. Otherwise, if the cosine of *shadowdirection* is negative, then the pixels discarded are those located at a horizontal coordinate within the minimum eighth of the horizontal coordinates. The same logic is used to discard pixels based on the sine of *shadowdirection* and the vertical coordinates. In Figure 3.7, the green outlines describe the exterior boundaries of the shadow objects. The pixels highlighted in magenta are those that remain following the erosion steps described in this subsection.

Finally, it is time to search for building objects which overlap the remaining pixels in the shadow object. Rather than combining the shadow pixels from all the shadow objects and looking for overlaps with any in the set, it is more helpful to keep track of which building objects overlap shadow pixels from the same shadow object. Therefore, the shadow objects are compared for overlaps one at a time. For each shadow object, every single building object is checked for overlap with that particular shadow. To be exact, the exterior boundary of every single building object is checked. It is significantly cheaper computationally and just as effective to check only the boundary pixels for overlap as opposed to checking all of the pixels in the filled object. Furthermore, each building object is actually checked five times during this step. After each check, the set of shadow pixels is incrementally translated in the direction of *shadowdirection*. This sounds strange but it is actually the same concept as that used by [8] when they check for overlaps. Intuitively, a shadow edge does not actually overlap a building edge. In fact, they are adjacent to one another. Throw in the imprecise nature of finding these edges, and it becomes clear that steps must be taken in order to create the necessary overlaps. Reference [8] simply double

dilates both objects. That approach runs the risk of overcrowding and producing overlaps where they should not exist. That is why this approach focuses on the edge of interest in the shadow object, and leaves the building edges in their true locations. The quintuple translation of the shadow pixels essentially is the same as a double dilation applied to both objects, just in a more controlled direction. Translating the shadow pixels multiple times does introduce the possibility of picking up erroneous overlaps on the ends, but this is why the extremes are discarded in the previous paragraph.

A new building object, *shadowedbuildobj*, is created for each shadow object, containing all the pixels from any building objects which have a nonempty intersection with the respective set of shadow pixels. While the steps described in this subsection are being performed, the steps in the following subsection are being performed simultaneously, and the set of building objects contained within *shadowedbuildobj* also include those which are found in this second overlap search.

3.3.2 Erosion of Convex Image of Shadow Object and Overlap Search.

As mentioned above, the steps in this subsection are performed alongside the steps described in the previous subsection. While the method described above erodes the shadow object itself, this method instead erodes the convex image of the shadow object. The convex image of the shadow object is found by filling the convex hull of the shadow object. Then, the convex image is eroded once using the basic 3×3 structuring element of ones. MATLAB® has a built-in function which finds the convex hull of an object, and this function creates points along the hull with coordinates of type *double*. When the coordinates are rounded to integers, extra pixels can be picked up. This is why the basic erosion is applied right away. Then the structuring element(s) determined above using *shadowdirection* is used in the same way as described in the first paragraph of the previous subsection to erode the convex image. This result is eroded again the same way as described in the second paragraph of the previous subsection in order to creater sparser

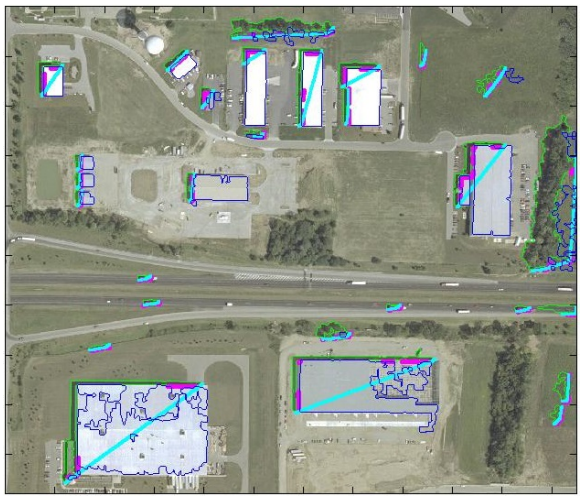

Figure 3.7: Illustration of how building pieces are found using the locations of shadows. The shadows are outlined in green. The shadow pixels which are used to search for overlapping building objects are highlighted in magenta and cyan. The magenta pixels are those which are found by eroding the shadow object. The cyan pixels are those which are found by eroding the convex hull of the shadow object. The blue outlines describe the building objects which are picked up during the overlap search. Original image data available from the U.S. Geological Survey [1].

line segments. The remaining pixels in the convex hull at this point do not need to be weeded any further. It is not necessary to remove the end points for this set of pixels, since they will not be translated during the overlap search. The set of shadow pixels in each convex hull remaining after these erosion steps is highlighted in cyan in Figure 3.7.

The translations described above as necessary to create overlaps between otherwise adjacent edges do not apply to this set of shadow pixels, which are taken from the convex hull. This is because they do not describe edges of the shadow object. Ideally the set cuts diagonally through the corresponding building. Intuitively and as illustrated in Figure 3.7, the convex hull erosion described here is most helpful when L-shaped shadows are

involved, but even for a rectangular shadow, it does not hurt to more thoroughly check for building overlaps. If the exterior boundary of a building object has a nonempty intersection with this set of shadow pixels, then the object pixels are added to the list in *shadowedbuildobj* for each respective shadow object.

3.3.3 Combining and Processing Building Objects.

The previous two subsections describe how building objects are combined into a single building object, *shadowedbuildobj*, representing the entire building which casts each shadow. However, there is actually another hoop to go through, which is left out in the earlier discussion, before each overlapping building object is added to *shadowedbuildobj* for the respective shadow. Even if a building object has a nonempty intersection with either of the two eroded shadow pixel sets, it can still be disregarded if it fails to meet one last condition. This condition compares the building object to the original shadow object outlined in green in Figure 3.7. If any part of the building object extends past the shadow in the wrong direction then it is not added to *shadowedbuildobj*. This relationship is determined by *shadowdirection*. If the cosine of *shadowdirection* is positive, then the minimum horizontal coordinate in the building object must be greater than the minimum horizontal coordinate in the shadow object. If the cosine of *shadowdirection* is negative, then the maximum horizontal coordinate in the building object must be less than the maximum horizontal coordinate in the shadow object. The same relationships must hold true for the sine of *shadowdirection* and the vertical coordinates in order to include the building object in *shadowedbuildobj* for that shadow. The building objects which make up *shadowedbuildobj* for each shadow are outlined in blue in Figure 3.7.

It is possible that a particular shadow object may not overlap with any viable building objects. If this happens then the corresponding *shadowedbuildobj* will be empty, and the shadow object will subsequently be discarded. This is not necessarily a bad thing, because

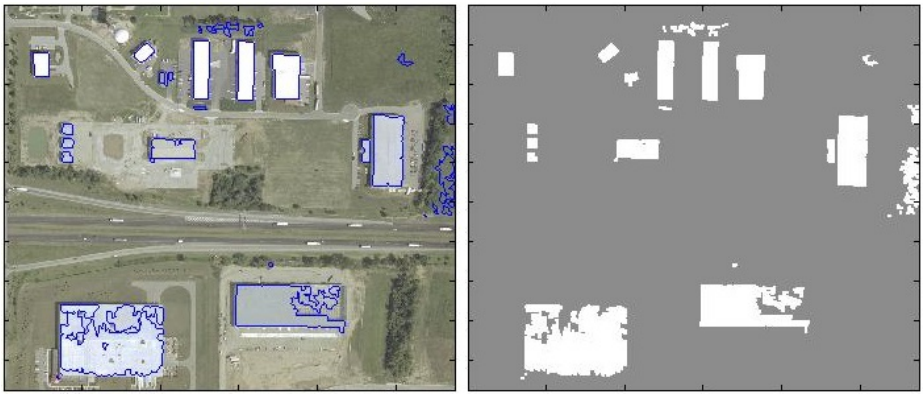

Figure 3.8: The building objects picked up during the overlap search are (a) outlined in blue, and (b) filled in for visualization. Original image data available from the U.S. Geological Survey [1].

it means the shadow object is most likely a false shadow or is cast by something other than a building.

The building objects which make up *shadowedbuildobj* for each shadow are once again outlined in blue, this time in Figure 3.8. Nothing has changed since Figure 3.7, but Figure 3.8 simply displays the outlines in a less cluttered manner. The adjacent image, which simply fills in the outlines of the building objects, can be compared to Figure 3.9.

Now that the segmented objects in the image are reassembled and grouped into potential bulldings, each grouping can be processed and analyzed for the likelihood of this potential. Toward that end, each *shadowedbuildobj* is looked at individually. At this point, each *shadowedbuildobj* is a set of pixels which belong to multiple separate objects which lie in the same vicinity as one another. Before the overall shape of the building object can be analyzed, however, the pixels must connect to form one single object. Therefore, the multiple separate objects are made to connect to one another through a series of morphological operations. Since each *shadowedbuildobj* is processed separately, there is no danger of making unwanted connections among the *shadowedbuildobj*s

belonging to different shadows. It is still important to avoid overdoing the morphological "closing," though. Generally, if it takes more than a handful of dilations to create connections among the separate objects in *shadowedbuildobj*, then the objects are not actually parts of a building. The series used in this algorithm is "dilate," "close," "close," "erode," "clean," but this is not the magical recipe for creating a building. It is simply an effective technique for combining the objects which does not tend to create false buildings. Then the exterior boundary of the connected pixels is traced to find the new *shadowedbuildobj*. If any of the pixels within *shadowedbuildobj* are unable to connect with the others, this tracing step will produce multiple objects once again. If this happens, only the object with the largest perimeter is retained as the new *shadowedbuildobj*.

The new *shadowedbuildobj* is then analyzed in terms of area. Usually, a building is larger than the shadow it casts, as long as the Sun is not positioned too far at an angle. Most useful satellite images are taken on clear, sunny days when the Sun is close to directly overhead. Still, for the sake of generality, a conservative condition is used in this step. A *shadowedbuildobj* is discarded only if it has a pixel area smaller than half the pixel area of the corresponding shadow object. The shadow object used for comparison here is the original object, which is outlined in green in Figure 3.7, not the eroded pixel sets. The remaining *shadowedbuildobj*s are displayed in Figure 3.9 and can be compared to the previous *shadowbuildobj*s before combination and analysis, which are displayed in Figure 3.8.

3.3.4 Final Processing to Complete Building Extraction.

Finally, each remaining *shadowedbuildobj* is analyzed using the following three criteria. A *shadowedbuildobj* is discarded if it has an extent less than 0.7, if it has an area less than 0.7 times the minimum expected building area estimate, or if its perimeter is less than half the minimum expected building perimeter estimate. Extent is once again defined as the ratio of the pixel area of the object to the pixel area of the filled convex hull of the

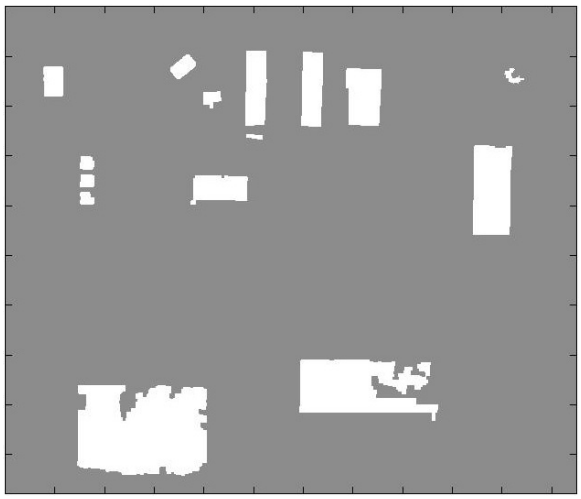

Figure 3.9: The building objects found during the overlap search have been assembed into individual buildings and connected. Original image data available from the U.S. Geological Survey [1].

object. This criterion is helpful in weeding out the more organically shaped objects which may still remain but are unlikely to represent buildings, such as those which actually represent forested regions. The minimum expected building area estimate is found by dividing the minimum expected building perimeter estimate by 4 and squaring it. These three criteria keep with the conservative nature which applies to the other discard criteria used throughout this process. The final set of extracted building objects is outlined in blue in Figure 3.10, but the algorithm is not quite finished yet. The blue outlines are still somewhat blob-like with missing chunks and jagged edges. To remedy this, it is assumed that the buildings are all rectangular. A rectangle of best fit is found for each building object border and outlined in magenta in Figure 3.10. The magenta rectangles are the final result of the building extraction algorithm, but since finding a best-fit rectangle is a nontrivial process, an explanation of this step is included in the next subsection.

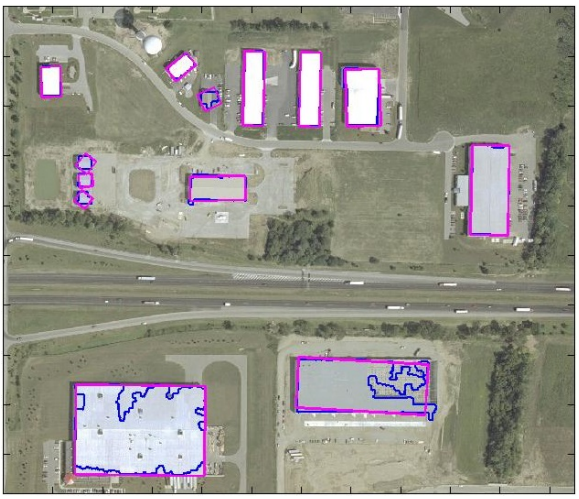

Figure 3.10: Results of the building extraction algorithm are shown in blue and magenta. The raw building objects are outlined in blue, and the rectangles of best fit which describe the final extracted building locations are outlined in magenta. Original image data available from the U.S. Geological Survey [1].

3.3.5 Finding a Rectangle of Best Fit.

The first step to finding a rectangle of best fit for the border of a region of pixels is to subtract the centroid location from each pixel location in the border. This translates the region so that the new centroid of the region is located at the origin of the Cartesian coordinate system. Then, the orientation, θ, of the region is determined to be the angle between the x-axis in the Cartesian coordinate system and the major axis of the ellipse which has the same second-moments as the region. Using θ, the region is temporarily rotated about its centroid, which has been temporarily translated to the origin, so that the orientation of the region becomes zero.

This temporary rotation is accomplished by applying the rotation to the entire set of pixels in the region border. For ease of communication, the following rotation equations refer to the rotation of a single pixel, (x_0, y_0), around the origin. The basic concept of the

rotation involves subtracting θ from the angle of the vector pointing to the pixel in order to produce a new angle, θ_{temp}, as described by

$$\theta_{temp} = \arctan\left(\frac{y_0}{x_0}\right) - \theta. \tag{3.2}$$

The vector with angle θ_{temp} pointing towards the pixel position after rotation, (x, y), has a length equal to the length of the original vector, which is equal to $\sqrt{x_0^2 + y_0^2}$. These temporary coordinates are therefore defined as

$$\begin{aligned} x &= \sqrt{x_0^2 + y_0^2}\cos(\theta_{temp}) \\ y &= \sqrt{x_0^2 + y_0^2}\sin(\theta_{temp}) \end{aligned}. \tag{3.3}$$

Equation (3.2) is substituted into Equation (3.3). Since $\cos(u \pm v) = \cos(u)\cos(v) \mp \sin(u)\sin(v)$, and $\sin(u \pm v) = \sin(u)\cos(v) \pm \cos(u)\sin(v)$, the new equations become

$$\begin{aligned} x &= \sqrt{x_0^2 + y_0^2}(\cos(\arctan(\tfrac{y_0}{x_0}))\cos(\theta) + \sin(\arctan(\tfrac{y_0}{x_0}))\sin(\theta)) \\ y &= \sqrt{x_0^2 + y_0^2}(\sin(\arctan(\tfrac{y_0}{x_0}))\cos(\theta) - \cos(\arctan(\tfrac{y_0}{x_0}))\sin(\theta)) \end{aligned}. \tag{3.4}$$

After applying more trigonometric identities and distributing $\sqrt{x_0^2 + y_0^2}$, Equation (3.5) simplifies to

$$\begin{aligned} x &= x_0\cos(\theta) + y_0\sin(\theta) \\ y &= y_0\cos(\theta) - x_0\sin(\theta) \end{aligned}. \tag{3.5}$$

Once the temporary rotation is applied to the set of pixels in the region border, it is much simpler to find the coordinates of the corner points of the best-fit rectangle. These corner points are referred to as $topleft$, $topright$, $bottomright$, and $bottomleft$. Although these labels seem arbitrary when the rectangle is later rotated back to its original orientation, they are helpful for the intermediate calculations. The top coordinate refers to the maximum y value in the rectangle. This value is found by clustering all of the positive

y coordinate values into 3 clusters. The mode of the set of pixels belonging to the cluster with the highest mean becomes the top value. This process sounds convoluted, but it makes sense intuitively. There is a cluster of redundant y values along the top edge of an ideal rectangle. The same is true for the bottom edge, as well as for the x values along the left and right edges. The same process used to find the top coordinate value is therefore adapted and applied toward finding the $bottom$, $left$, and $right$ coordinate values. Since top and $bottom$ are y coordinates, and $left$ and $right$ are x coordinates, these coordinates can be combined and substituted into Equation (3.5) to find the corner points and reverse the rotation so that the orientation of the best-fit rectangle matches the orientation of the region. Once these corner points are calculated, the original centroid coordinates of the region are added to the corner points to translate the best-fit rectangle back to where it belongs.

The corner points are all that is needed to define the best-fit rectangle, but it is also helpful to find the angles of the vectors normal to the walls of the rectangle. The angle of the normal vector for the "top" wall is equal to $\theta + 90°$. The vector normal to the "right" wall has an angle equal to θ, and the "bottom" wall has a normal vector with angle equal to $\theta - 90°$. If θ is positive, the angle of the vector normal to the "left" wall is equal to $\theta - 180°$, but if θ is negative, the angle is equal to $\theta + 180°$.

3.4 Checking Lines of Sight for Obstructions

The remainder of this chapter utilizes information garnered from the building extraction algorithm described above, which can be applied to any image. The goal of the remainder of this chapter is to use the building locations discovered by the building extraction algorithm to improve geolocation accuracy. Since the aerial RGB images used to validate the building extraction algorithm are large and complex, the images are scaled down before they are used to validate the remainder of the research method. This scaling decreases the computation time exponentially. In order to have results near real-time, the

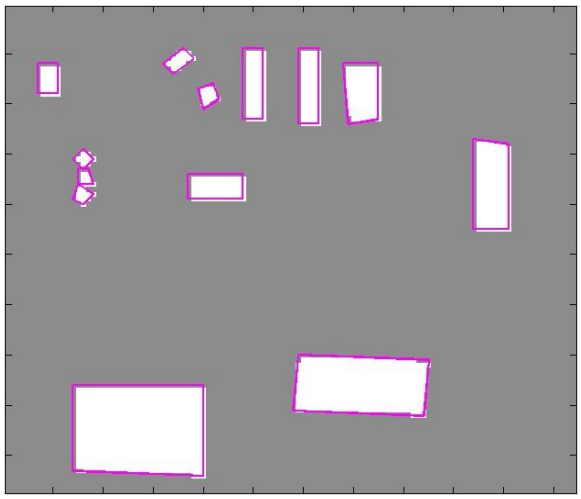

Figure 3.11: Results of the building extraction algorithm are scaled to 1% of the original size. White logic 1 pixels make up the buildings, which are outlined in magenta. Original image data available from the U.S. Geological Survey [1].

image should be scaled to about 100×100 pixels. All that is needed from the bulding extraction algorithm in order to improve geolocation in the desired manner are the resulting rectangles of best fit, described by their corner points and wall normal vectors. Therefore, to scale down the original image of size 977×1149 pixels in Figure 3.1, all that is required is to divide the coordinates of the rectangles each by 10. The other color data from the original image can be discarded. The resulting image is shown in Figure 3.11. The original image used for Figure 3.11 has a ground sample distance (GSD) of 0.6096m. Therefore, the distance on the ground represented by one pixel in the scaled version shown in Figure 3.11 is 6.096m. There is a slight distortion due to the rounding required to create integer coordinates following the division. Also, since the buildings are made up of fewer pixels, some of the tilted buildings appear to have jagged edges.

The next step involves determining which sensors have a direct LOS to the transmitter and which have an obstructed LOS. This step is incorporated into the process for finding

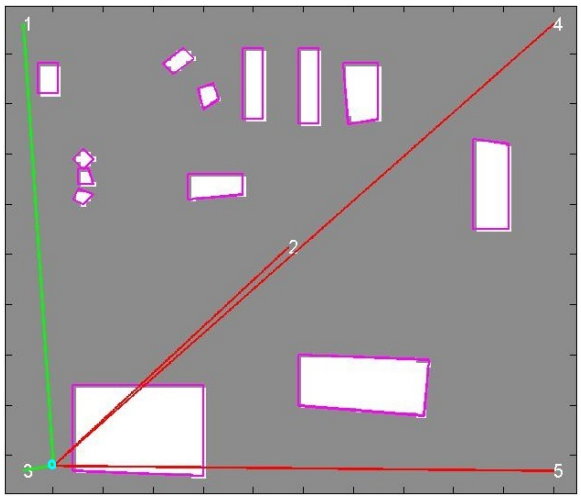

Figure 3.12: Obstructed lines of sight are drawn in red, while others are drawn in green. The arbitrary transmitter location is plotted as a cyan circle, and the sensor locations are numbered in white. Original image data available from the U.S. Geological Survey [1].

the shortest paths in Section 3.5 and is implemented multiple times. In Section 3.5 it is referred to simply as "checking for obstructions." In this section, this step is described as if the transmitter location is known. Under these conditions, this is a very simple step. A line is drawn from the transmitter location to each of the sensor locations. If any of these lines crosses any pixels which are logic 1 pixels and therefore belong to a building, then the offending line describes an obstructed LOS. The sensor belonging to this line is therefore obstructed. The lines of sight for an arbitrary transmitter location and sensor configuration are shown in Figure 3.12. The arbitrary transmitter location and sensor configuration shown are used exclusively in the remainder of this chapter to illustrate the processing steps, and were simply chosen for fluidity and ease of visualization. As discussed in subsequent chapters, the algorithm can be applied to any transmitter location and any sensor configuration.

3.5 Finding the Shortest Paths

Like Section 3.4, this section also describes a step that will be performed multiple times in Section 3.6. It will be referred to simply as "finding the shortest path." The process is described in this section as if the transmitter location is known. The goal of this process is to find the shortest distance travelled by the signal from the transmitter to reach each sensor. If the sensor has a direct LOS to the transmitter, then this problem is simple. The LOS is the shortest path. However, if a sensor does not have a direct LOS, then the received signal is actually a reflection of the transmitted signal off one or more building walls. It is assumed for simplicity that if there is no path available to the signal which involves 2 or fewer reflections, then the received signal is too weak to be detected by the sensor. If this case occurs, the offending sensor remains obstructed. The implications of this event are discussed in Section 3.6.

In order to find the shortest paths, the sensors must be checked for obstructions. The path length of the signal for each of the unobstructed sensors is simply the Euclidean distance between the sensor location and the transmitter location. For the obstructed sensors, it is trickier. Eventually each sensor will be looked at individually to find its shortest path, but first there are a few computations which apply to the path search for all the sensors and can be made up front. Any reflective path a signal takes will have to first reflect off of a building facing the transmitter. To find all possible paths, all of these potential reflection points must first be found.

Walls are listed by the building to which they belong. Therefore, each building is looked at individually to find which of its four walls, if any, face(s) the transmitter with an unobstructed view. A new list is made, called *visible*, which includes all the walls which are visible to the transmitter, indexed by building. To determine whether a wall qualifies for inclusion in this list, its orientation and position are evaluated. When the rectangle of best fit is found, the angle of the normal vector for each wall is also found. The normal

vector for each wall is displayed in Figure 3.13 as a cyan arrow. This normal vector is essential in determining whether the wall faces the transmitter. The wall is determined to be facing the transmitter if and only if the transmitter lies in the region bounded by the line running through the points in the wall and on the same side of the wall as its normal vector. This determination is described by Equation (3.6), which is the same as Equation (2.11) in *Chapter II*. This concept is also illustrated in Figure 2.10 in *Chapter II*. Just as described in Section 2.5 of *Chapter II*, $\vec{a}$ refers to the transmitter location, $\vec{r}_w$ refers to a point in the wall, and $\hat{\Omega}_w$ refers to the normal vector of the wall.

$$(\vec{a} - \vec{r}_w) \cdot \hat{\Omega}_w > 0 \tag{3.6}$$

If the wall is determined to be facing the transmitter, the LOS between the transmitter and the wall is checked for obstructions. This is a little computation-intensive, since every point along the wall must be checked. If any point along the wall facing the transmitter has an unobstructed LOS, the wall is added to *visible*. However, when the wall is added to *visible*, only the unobstructed points in the wall are retained.

Once the list of walls which can potentially serve as an initial reflection point has been compiled, a second list of walls, called *visible_double*, is compiled to include all walls which can potentially serve as a second reflection point. Compilation of this list begins with looking at each wall in *visible*. Each wall in *visible* is compared to every other wall which does not belong to the same building as that wall. Ignoring the walls belonging to the same building saves a little time since it can be assumed that no walls in a rectangular building can possibly face one another. To determine if two walls face one another, Equation (3.6) must be applied to each wall. Instead of $\vec{a}$ referring to the transmitter position, in this application it refers to a point in the opposite wall. If both walls are found to face one another, then every point in the potential second reflection wall is checked against every point in the *visible* wall for obstructions. If the potential second reflection

wall and the *visible* wall face one another with an unobstructed view, then the two walls are added to *visible_double* as a pair. When these two walls are added to *visible_double*, only the points with an unobstructed view of at least one point on the opposing wall are retained. Lumping together the unobstructed points for each wall is an oversimplication, since there are cases when walls are partially obstructed from one another, and the unobstructed portions may not entirely match up, but the benefits of this simplication outweigh the risk of inaccuracy.

At this point, two lists have been created. When the transmitter is assumed to be the starting location, the walls in *visible* describe the first half of all potential single-reflection paths. Likewise, the walls in *visible_double* describe the first two-thirds of all potential double-reflection paths. Another list can be created in a similar fashion to describe triple-reflection paths, but this is unnecessary, since the signal loses power with each reflection. Three reflections would make the signal too weak for detection by the sensors.

Now, it is finally time to look at each of the obstructed sensors. Each obstructed sensor must be checked against every wall in *visible* and every second reflection wall in *visible_double*. If the wall faces the sensor with an unobstructed view, then that *visible* wall or *visible_double* pair of walls is added to the list *paths* for that sensor. If the obstructed sensor cannot complete any paths which were started in *visible* or in *visible_double*, then that sensor remains obstructed. Once each initially obstructed sensor is compared against each partial path, the list of possible signal paths, *paths*, for each sensor is complete. Then, *paths* for each sensor can be evaluated to find the shortest path from the transmitter to that sensor. Up until now, reflection "points" are stored as pieces of building walls in either *visible* or *visible_double*. To find the optimum point of reflection on the wall, each point in the wall piece is evaluated in order to find which point creates the shortest path in the context of the other reflection points in the path. Since it is assumed that a wall exhibits Lambertian reflectance, the Law of Reflection is irrelevant.

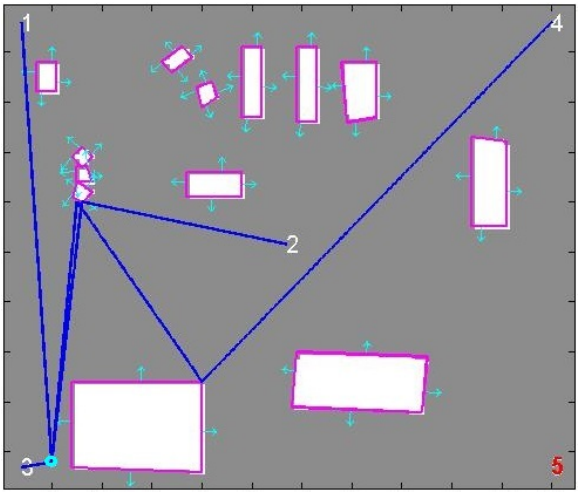

Figure 3.13: The shortest path from the transmitter to each sensor is drawn in blue. The sensor which remains obstructed even after searching for a single- or double-reflection path is highlighted in red. The transmitter location is plotted as a cyan circle. The buildings are outlined in magenta, and the normal vectors for each wall of each building are displayed as cyan arrows. Original image data available from the U.S. Geological Survey [1].

Once the point of reflection is found for each wall piece in regards to the path it describes, the length of that path is calculated as the sum of the distances from one point to the next along the path, beginning with the transmitter and ending with the sensor. The shortest paths from an arbitrary transmitter location to each sensor in an arbitrary configuration are illustrated in Figure 3.13.

3.6 Combining Signal Information with Image Information

This section describes how all of the previously discussed steps can be incorporated with TDOA in order to improve geolocation. The TOA of the signal at each sensor is directly proportional to the distance the signal travels from the sensor to the transmitter. Since the transmitter being located has an unknown position, this distance as well as the time it takes the signal to travel from the transmitter to the sensor is unknown. However,

the TDOA is known. If the signal has to travel farther to reach sensor i than it does to reach sensor j, then sensor j will detect the signal before sensor i detects the signal. The difference in this timing can give a clue about how much farther the signal travels to reach sensor i than it does to reach sensor j. With traditional TDOA geolocation, the timing information is utilized under the assumption that the signal travels directly from the transmitter to the sensor with no reflections. This thesis does not apply this assumption. This section describes how TDOA can be used to determine the location of a transmitter under the assumption that the signal may be reflected up to two times before reaching a sensor.

How is the TDOA data collected? In this research, it is not. Collecting TDOA data is about as straightforward as it sounds with the correct equipment. However, sample TDOA data is not available for the aerial RGB images used in this research. Therefore, the TDOA data is simulated. A transmitter location is arbitrarily chosen to be the actual location, and the shortest paths from this location to the sensors are determined. These path lengths along with the GSD are used to calculate the simulated actual TOA of the signal at each sensor, using $c/1.000293$ as the propagation speed. The refractive index of air is 1.000293, and c is the speed of light in a vacuum. If a sensor is determined to be obstructed with no available single- or double-reflection paths, then it is assigned an infinite TOA value. Then, additive white Gaussian noise (AWGN) is added to the TOA data, and the simulated TDOA values for all the sensors are calculated in reference to the same sensor. Any sensor can be chosen to be the reference sensor as long as it does not have an infinite TOA value. If a sensor has an infinite TOA value, then it also has an infinite TDOA value. The simulated transmitter location is plotted as a cyan circle, and the shortest paths are drawn as dashed blue lines in Figure 3.14.

In order to find the MLE of the transmitter location, a grid search is performed to determine what the noiseless TDOA values would be if the transmitter were located at

each grid position. Then, these theoretical values can be compared to the simulated values from the "actual" transmitter location. This comparison is accomplished using a measurement referred to here as $diff$, which is the average across the sensors of the square of the difference between the theoretical TDOA value for a sensor and that sensor's simulated actual TDOA value. The MLE of the transmitter location is determined to be the grid location which produces the smallest $diff$. If one of the sensors is completely obstructed from a grid location, then its theoretical TDOA value is infinite. These infinite values are ignored during the $diff$ computation. During the grid search, all sensor locations as well as locations which lie within the buildings or within 1 pixel of the building boundaries are ignored as possible transmitter locations, because it is assumed that the transmitter is not in any of these locations.

When performing a grid search, shortest paths are calculated for many possible transmitter locations. For all of these possible transmitter locations, however, the sensor locations remain the same. Therefore, flipping the method of finding the shortest paths saves a significant amount of time. The method is flipped by using the sensor locations as starting points for the paths, rather than using the transmitter location as the starting point for each path. This does not affect the accuracy of the results. It simply means that the final half to two-thirds of the path between a transmitter and a sensor is calculated before the beginning portion is calculated. This allows the *visible* and *visible_double* wall lists to be created and saved outside of the grid search instead of being recomputing for every grid search guess. Each grid search guess can then be compared to the path portions already created. Of the paths it is capable of completing, the shortest one is used for that grid location. Switching up the path calculation in this way speeds up the process roughly by a factor of 20. Populating the *visible* and *visible_double* wall lists is considered an "offline" task, since it can be accomplished without the live TDOA data. These lists are constant for a given image region and sensor configuration, even if the transmitter is moving.

Even though the original aerial RGB image has already been scaled down for this geolocation portion, and even though populating the *visible* and *visible_double* lists offline saves time, a grid search for the MLE is still very computation-intensive. For an image the size of Figure 3.11 and with the number of walls present in the image, it would take around 1 hour to find the MLE of the transmitter location while checking every possible grid location. If, however, grid locations are checked in increments of 10 pixels, this computation time reduces to about 1 minute. Depending on the real-time requirements of the implementation, it may or may not be necessary to check every single grid location. However, since this research aims to produce a high volume of results for comparison, it is necessary to reduce the time required to find each result.

There is another option to reduce computation time which is a compromise between checking every grid location and checking every tenth grid location. This option involves first applying the incremental method and then using the result of that method to decrease the grid search area. If increments of ten pixels are used, then the result of the incremental search lies in the center of a 10×10 pixel region whose other points have not been searched. Searching each one of these other points can potentially yield a more accurate result. Under very noisy conditions, the result of the incremental search may point to a region which does not contain the true value. In this case, the extra fine-resolution search will not increase the accuracy of the result. However, depending on the situation, the finer measurement resolution gained by the extra search may be worth the wait. Theoretically, adding this extra search to the incremental search should double the computation time from roughly 1 minute to roughly 2 minutes, since the number of grid points being checked is doubled. However, the added time may vary significantly depending on the region found in the incremental search. The added time may be shorter than expected if the grid points in the region found by the incremental search each produce *visible* and *visible_double* wall lists that are much shorter than the average for the image. Shorter wall

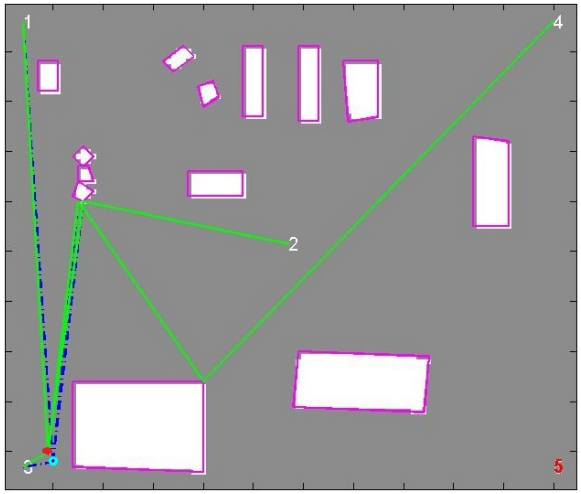

Figure 3.14: The MLE of the transmitter location computed from a grid search with coordinate increments of 10 is plotted as a red star, while the "actual" simulated transmitter location is plotted as a cyan circle. The obstructed sensor is highlighted in red, while the other sensors are shown in white. The buildings are outlined in magenta, and the shortest paths from the MLE to the sensors are drawn in green. The shortest paths from the actual transmitter to the sensors are drawn as dashed blue lines. If the dashed blue lines are difficult to see, it is because the green lines are drawn over top of them. Original image data available from the U.S. Geological Survey [1].

lists means fewer paths and path branches to check in order to find the shortest paths. Figure 3.14 shows the MLE transmitter location as a red star, found using grid coordinate increments of 10, followed by a fine-resolution search within the resulting 10×10 region. The shortest paths to this location are drawn in green, and the obstructed sensor is highlighted in red.

There may be another way to reduce the time requried to find each result, which is not used in this thesis. As described above, the *visible* and *visible_double* wall lists are populated offline, and the grid search to complete the paths and calculate the theoretical TDOA values at each grid location is performed online in this thesis. However, the

theoretical TDOA values for the grid locations can actually be calculated and saved offline as well with a few modifications. In this thesis, an incremental search is performed followed by a fine-resolution search based on the MLE found during the incremental search. Choosing the region on which to perform the fine-resolution search requires live TDOA data. However, if every single location in the grid were checked, live TDOA data would not be required to complete the search. Only the comparison of each set of theoretical TDOA data in the grid to the set of measured TDOA data would need to be performed online. This may decrease the online time required to find the location of a transmitter. However, there are drawbacks, and the decrease in time may not be significant enough to be worth them. Even if the theoretical TDOA values are calculated offline, they would still have to be compared to the measured TDOA values online, and this will take time. Also, since the entire grid is searched, many more values are being compared in this method, so a reduction in time is not guaranteed.

The main reason why this thesis both calculates the theoretical TDOA values and performs the comparisons online is because saving all of the theoretical values offline to be compared at a separate time takes up a lot of memory. For a 100×100 image, enough memory must be allocated to save 10,000 sets of TDOA values. Each set must contain the TDOA values for all of the sensors in relation to that grid location. The other issue is the time it takes to search the entire grid instead of searching in increments. As stated earlier, checking every single location takes around 1 hour and usually longer depending on the image, the number of buildings, and the sensor configuration. This added length of time required for the offline calculations may not be an issue depending on the mission, and it may be worth the potential decrease in the online computation time. Especially if a transmitter needs to be located multiple times within an image region with a stationary sensor configuration, it is beneficial to push the time costs to the front in order to track the transmitter closer to real time.

IV. Results and Analysis

This chapter provides data collected from various simulations using the method of geolocation presented in this thesis. In each simulation, the result of this thesis method is compared to the result calculated from a current geolocation method of using the Taylor series to solve for the intersection of hyperbolic curves created from the simulated TDOA data. This current method is discussed in Section 2.4 of *Chapter II*, which is taken from reference [3].

4.1 Building Extraction Results

Before discussing the overall geolocation results, building extraction results are first provided for various orthorectified aerial RGB images in order to show the validity of the method. The following series of figures consists of five pairs of figures. Each pair shows the results from a particular image. The first figure in each pair shows the major steps in the building extraction process as explained in *Chapter III* as well as the runtime of the algorithm for that image. Remember, this building extraction portion is performed offline. The major steps from top left to bottom right are (a) the grayscale intensity image after contrast enhancement; (b) the processed shadow objects and the *shadowdirection* vector labelled in cyan; (c) the building objects in blue found to overlap the shadow objects; and (d) the result of the building extraction algorithm, with the building objects in blue and the rectangles of best fit in magenta. The second figure in each pair shows some of the possible variations in the results of the algorithm. Due to the randomness in the clustering step, results may vary slightly for each image.

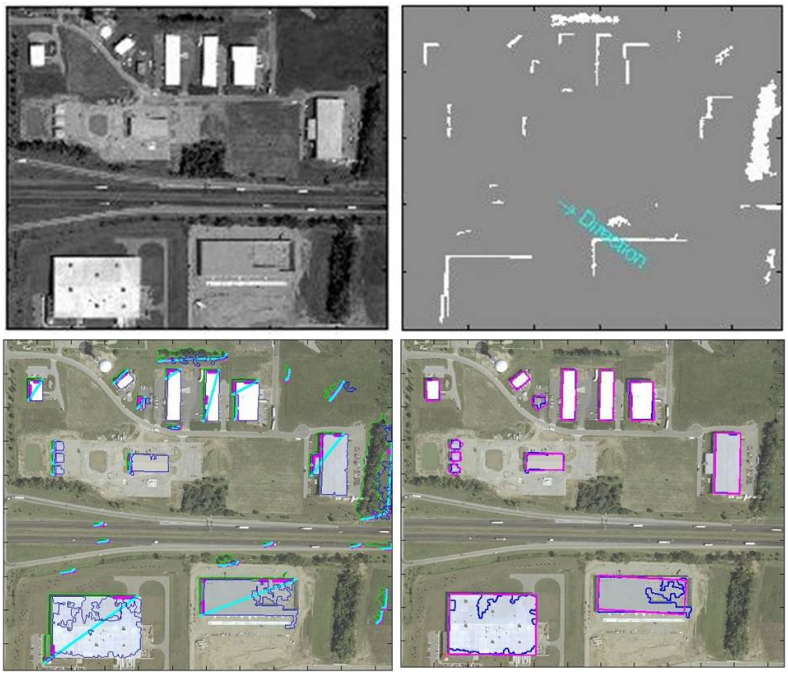

Figure 4.1: Results of the main steps in building extraction process for Image 1 (17.09 min). Original image data available from the U.S. Geological Survey [1].

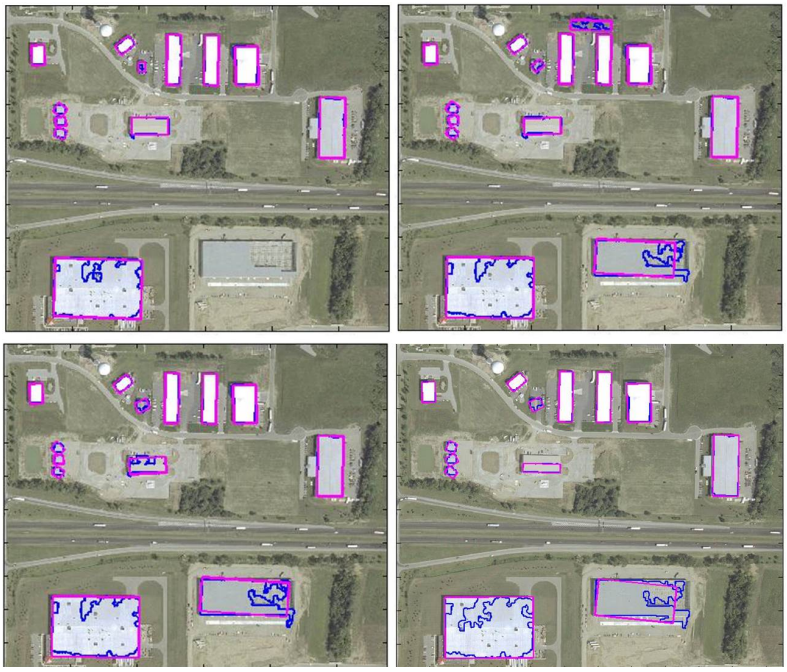

Figure 4.2: Sample variations in the result of the building extraction algorithm applied to Image 1. Original image data available from the U.S. Geological Survey [1].

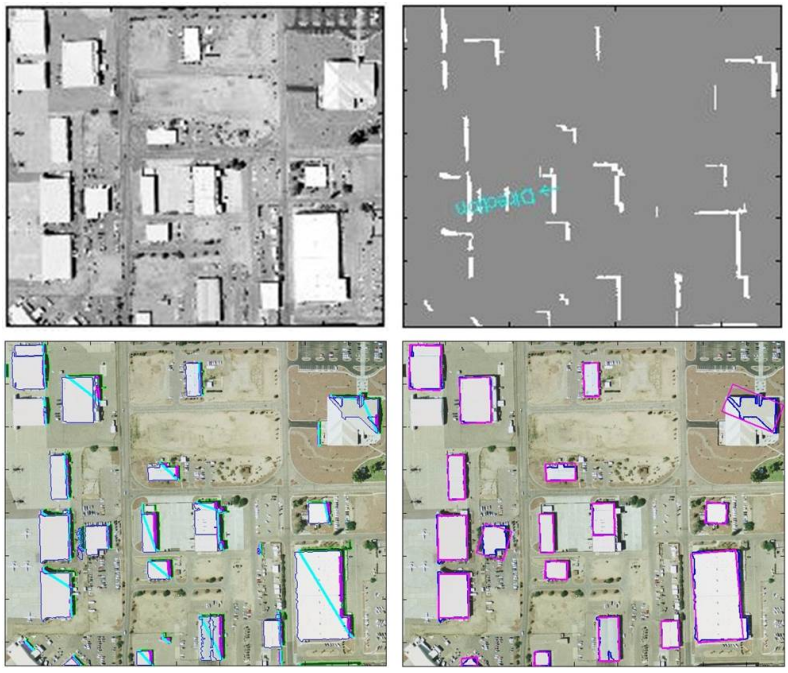

Figure 4.3: Results of the main steps in building extraction process for Image 2 (30.32 min). Original image data available from the U.S. Geological Survey [1].

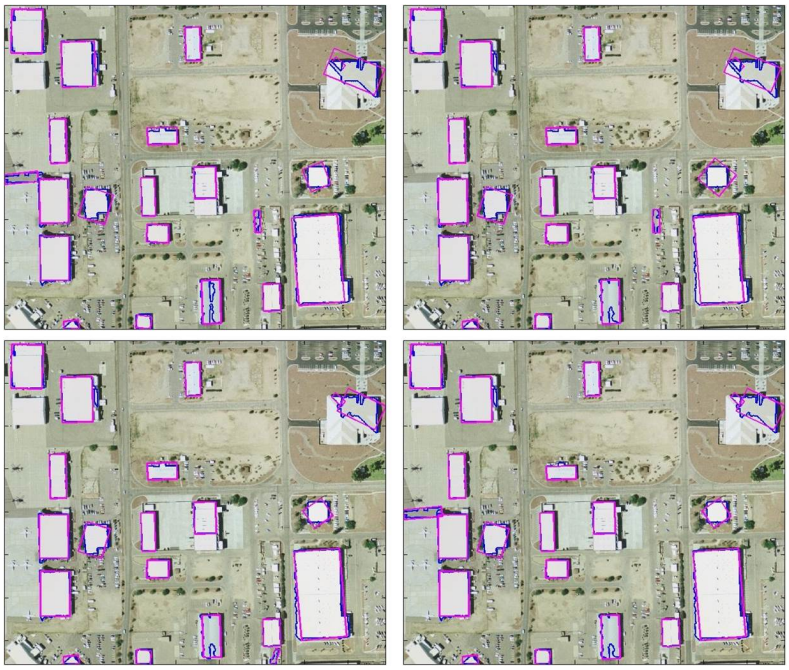

Figure 4.4: Sample variations in the result of the building extraction algorithm applied to Image 2. Original image data available from the U.S. Geological Survey [1].

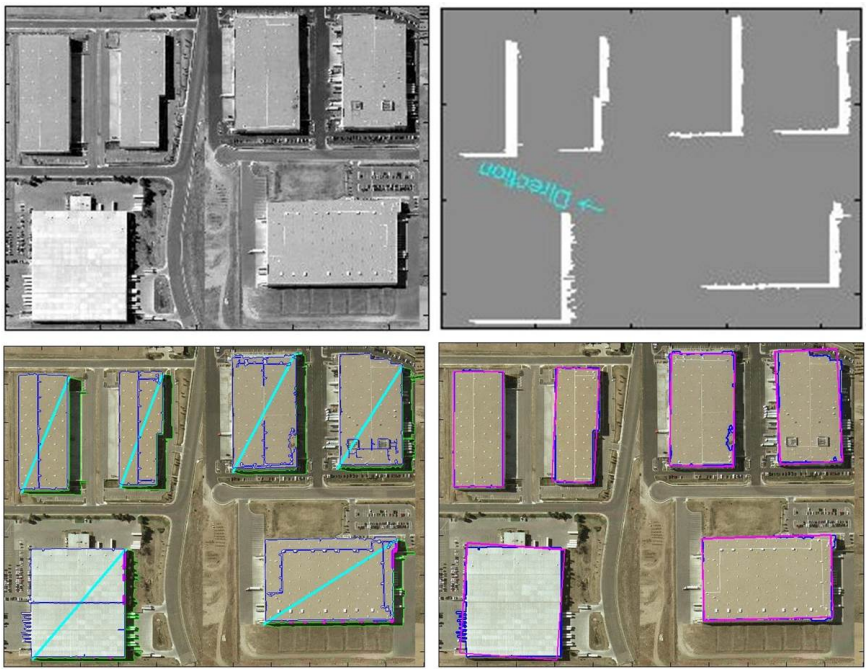

Figure 4.5: Results of the main steps in building extraction process for Image 3 (7.53 min). Original image data available from the U.S. Geological Survey [1].

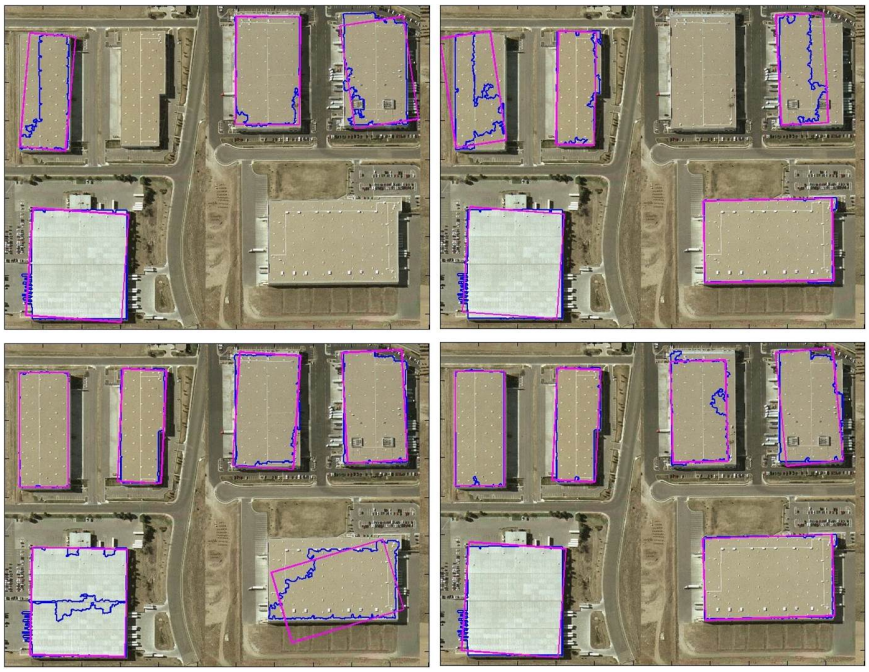

Figure 4.6: Sample variations in the result of the building extraction algorithm applied to Image 3. Original image data available from the U.S. Geological Survey [1].

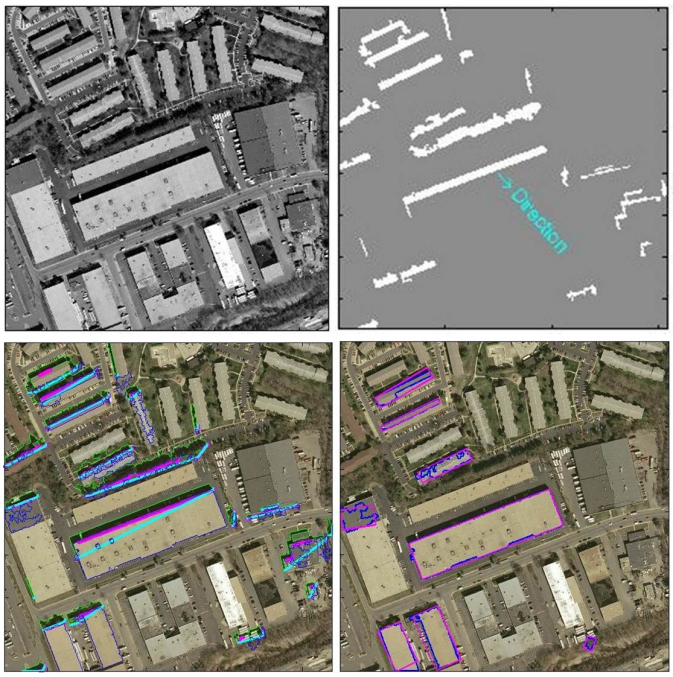

Figure 4.7: Results of the main steps in building extraction process for Image 4 (35.31 min). Original image data available from the U.S. Geological Survey [1].

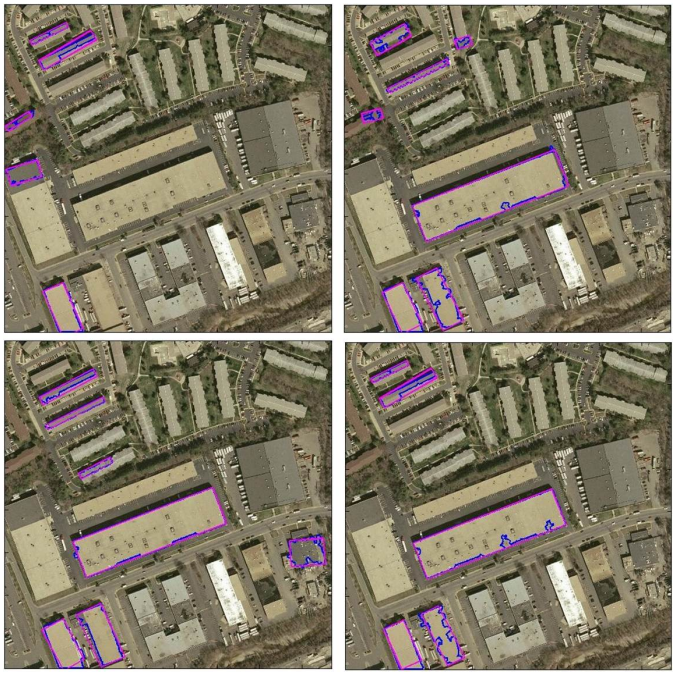

Figure 4.8: Sample variations in the result of the building extraction algorithm applied to Image 4. Original image data available from the U.S. Geological Survey [1].

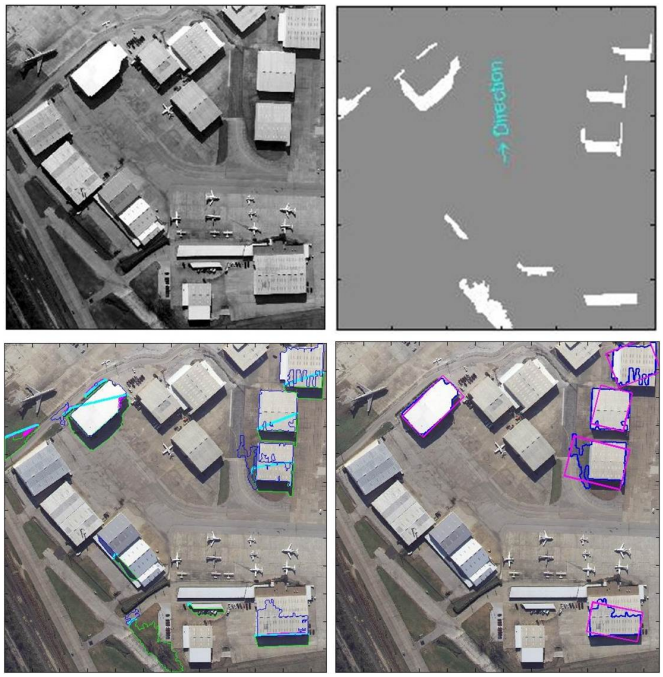

Figure 4.9: Results of the main steps in building extraction process for Image 5 (5.06 min). Original image data available from the U.S. Geological Survey [1].

Figure 4.10: Sample variations in the result of the building extraction algorithm applied to Image 5. Original image data available from the U.S. Geological Survey [1].

The building extraction algorithm yields effective results when applied to the first three images depicted in Figure 4.1 through Figure 4.6. However, Image 4 and Image 5, depicted in Figure 4.7 through Figure 4.10 appear to be poor cases for the algorithm. In the case of Image 4, difficulty arises from how busy the image is. Almost all of the buildings are polychromatic, and they are all oriented in a way that only one side of each building overlaps its shadow. Also, many of the buildings have jagged borders which confuse the algorithm especially during the shadow processing stage, since it is tailored to weed out jagged treelines. Image 5 is a poor case due to a couple factors as well. Many of the buildings are not only polychromatic, but are also made up of many small details which break the building object into many tiny pieces. On top of this issue, some of the buildings are very similar in color to the surrounding area, and when the edges are not well-defined, clustering groups these pixels into one large object. The other factor is the distribution of the shadows. Some of the buildings in Image 5 are clumped together so tightly that their shadows connect to one another. When two shadows connect, the algorithm treats them as one shadow object, and in the case of Image 5 these combined shadows are discarded for being too large.

The building extraction algorithm developed in this thesis is a compilation of techniques already used in current building extraction algorithms, many of which are discussed in *Chapter II*. These current algorithms are more sophisticated than the one used in this thesis. The algorithm used here was developed simply to show that it is possible to find the location of building walls in order to use them to improve geolocation. Even if the extracted building walls do not align perfectly with the actual building walls, a small tilt or position error is acceptable. This is due to the assumption made earlier which assumes that building walls have Lambertian reflectance. If signal reflections off of the walls had to obey the law of reflection, then the signal would only be able to travel in one direction after hitting the wall, and any small tilt in the wall position would change this

direction. However, with Lambertian reflectance, the signal bounces of the wall in all directions, which means that the exact angle of the wall is much less relevant.

4.2 Geolocation Results

In this section, the extracted building locations from the images shown in Section 4.1 are incorporated into a geolocation algorithm as described in Section 3.6 of *Chapter III*. The results of various simulations are shown below. For each setup, The simulation is run fifty times, and the average error and runtime of these fifty simulations are recorded both for this thesis method and for the current Taylor series method, which iteratively solves for the intersection of hyperbolic curves created from the TDOA data. The Taylor series method requires an initial position guess, but the placement of this guess does not affect the result as long as there are no local minima. Therefore, the same initial position guess is used for the Taylor series method throughout the simulations. Position locations throughout this chapter are described by pixels in relation to an origin located at the top left corner of the image. The y direction increases positively when travelling vertically downward from the origin.

Furthermore, the runtime for this thesis method is divided into an offline runtime and an online runtime. The offline runtime includes the time it takes for the building extraction algorithm to extract the building locations for the particular image. The offline runtime also includes the time it takes to populate the *visible* and *visible_double* wall lists for a given sensor configuration and set of building locations. Populating the wall lists usually takes less than half of a minute, but every bit of time that can be saved helps. The online runtime is the time it takes to search for the transmitter location in the manner described in Section 3.6 of *Chapter III*. This search time includes the time it takes to find the shortest path to each guess by completing the wall lists and the time it takes to compare the theoretical TDOA values to the actual TDOA values in order to find the MLE of the transmitter location. The offline operations can be performed at any time and only need to

Table 4.1: Statistics for Extracted Buildings

Image	Orig. Size (px)	GSD (m)	Sc. Factor	Sc. Size	GSD	Extraction (min)
1	977×1149	0.61	10	98×115	6.1	17.09
2	1509×1777	0.3	15	101×119	4.5	30.32
3	1677×2201	0.3	15	112×147	4.5	7.53

Table 4.2: Comparison of Image Geolocation Averaged over 50 Simulations for Thesis (T) and Current Taylor Series (C) Methods

Image	T Error (m)	C Error (m)	T Offline (min)	T Online (s)	C Runtime (s)
1	23.73	48.88	17.45	68.25	0.0065
2	30.35	66.45	30.85	134.42	0.022
3	94.24	143.14	7.90	41.28	0.063

be performed once for a given sensor configuration and image region, but the online portions must be performed with live TDOA data.

The first simulation compares the geolocation results for Images 1-3 from Figure 4.1, Figure 4.3, and Figure 4.5. Table 4.1 describes the size in pixels and GSD in meters of the original and scaled versions of each image. Table 4.2 compares the error and runtime averaged over fifty simulations for both the thesis method (T) and the current method (C) with an AWGN standard deviation (σ_N) of 25m, an actual transmitter centrally located at $(75, 35)$, and a sensor configuration which places sensors in each of the four corners and in the center of each image. For the simulations in this chapter, the AWGN is a timing error added to the simulated TOA values for the actual transmitter location. Although the error is in time, it is reported here as a distance in meters for ease of comparison. Since a TOA

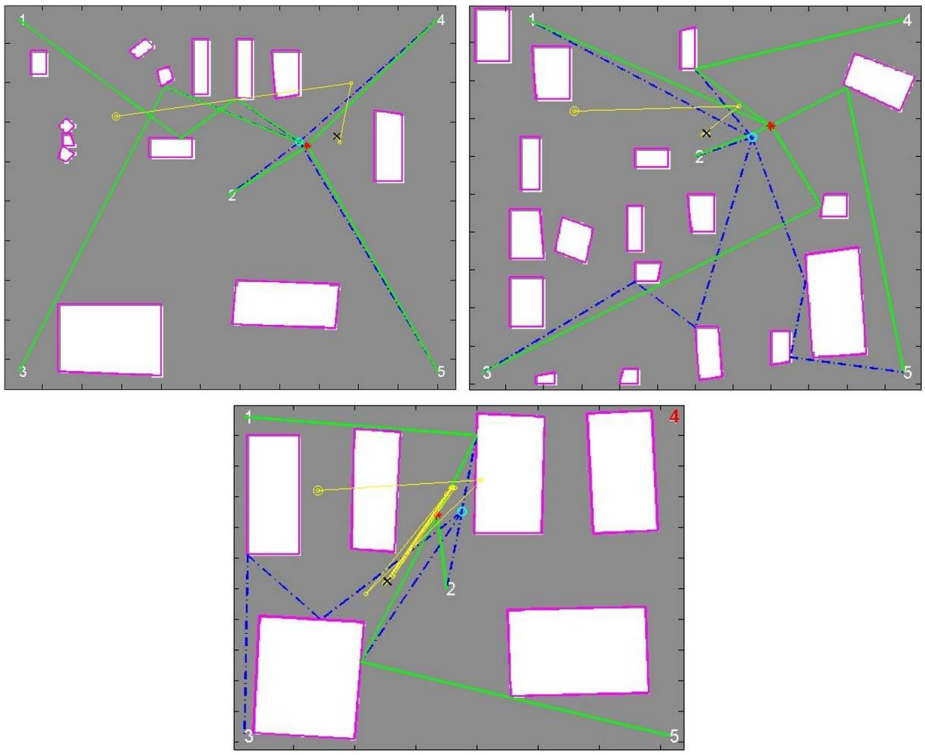

Figure 4.11: In each picture, the actual transmitter location is plotted as a cyan circle, and the shortest paths from the transmitter to the sensors are drawn as dashed blue lines. The result of this thesis algorithm is plotted as a red star, and the paths from this location to the sensors are drawn as green lines. Each iterative result of the current method is plotted as a yellow circle and connected to the next iterative result by a yellow line, in order to illustrate the trajectory of the iterations. The final result of the competing method is plotted as a black X. Simulation results are shown top left to bottom for (a) Image 1; (b) Image 2; (c) and Image 3. Original image data available from the U.S. Geological Survey [1].

error corresponds to a path-length error, stating the σ_N in terms of distance is appropriate. Figure 4.11 illustrates one of the fifty simulations for each of the images.

The second simulation compares the geolocation results for five different transmitter locations in Image 1 with a σ_N of 25m and with the same sensor configuration as shown in

Table 4.3: Comparison of Transmitter Locations Averaged over 50 Simulations for Thesis (T) and Current Taylor Series (C) Methods

	Tx Location	T Error (m)	C Error (m)	T Online (s)	C Runtime (s)
(a)	(10,92)	24.04	NaN	86.17	0.019
(b)	(75,35)	23.73	48.88	68.25	0.0065
(c)	(37,28)	23.82	41.10	122.40	0.0039
(d)	(82,85)	25.40	154.40	103.18	0.062
(e)	(91,50)	30.99	35.77	76.15	0.0068

Figure 4.11(a). These five transmitter locations were deliberately chosen to represent differing situations. These situations include situations where one or more sensors is obstructed, situations where the current method might be expected to perform more accurately, and situations where this thesis method might skip over the actual location due to the incremental nature of the grid search. For visualization, one simulation for each transmitter location is illustrated in Figure 4.12. The average values of the results are presented in Table 4.3, where "C" refers to the current method, and "T" refers to this thesis method. Since the sensor configuration has not changed, the offline runtime is the same for every transmitter location and therefore is not included. It is 17.45 minutes, the same value as the offline runtime in Table 4.2 for Image 1.

Figure 4.12 and Table 4.3 show that the current method does not converge to a result for transmitter location (a). The same nonconvergence issue arises in some of the following simulations as well. In these cases, the differences in TDOA values create curves that do not intersect in real space. This is a complex idea, but intuitively if TDOA values are unequally distorted (as caused by reflection paths), then it is possible that they will not intersect in one single, reasonable location. This lack of covergence is a downfall of the particular method of solving the hyperbolic equations. There are other current

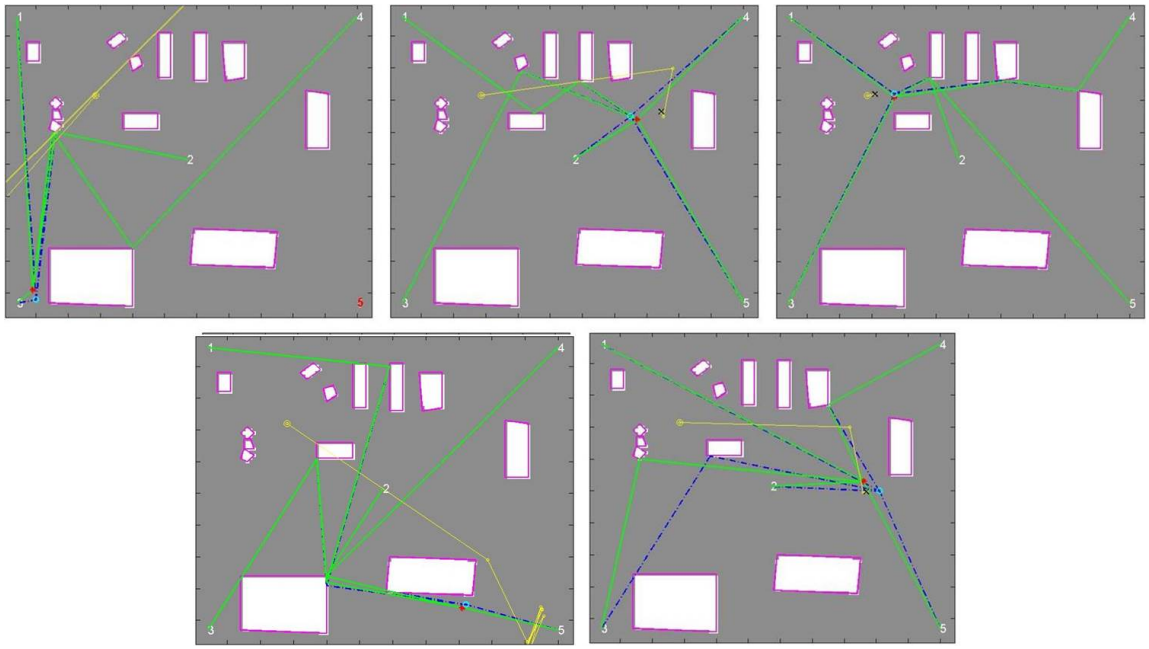

Figure 4.12: In each picture, the actual transmitter location is plotted as a cyan circle, and the actual shortest paths from the transmitter to the sensors are drawn as dashed blue lines. The result of this thesis algorithm is plotted as a red star, and the paths from this location to the sensors are drawn as green lines. Each iterative result of the current method is plotted as a yellow circle and connected to the next iterative result by a yellow line, in order to illustrate the trajectory of the iterations. The final result of the competing method is plotted as a black X. Simulation results are shown from top left to bottom right for transmitter locations (a) $(10, 92)$; (b) $(75, 35)$; (c) $(37, 28)$; (d) $(82, 85)$; and (e) $(91, 50)$. Original image data available from the U.S. Geological Survey [1].

methods of solving these equations, which do not have this convergence issue. Some of these methods are discussed in [3], but the iterative method used here is one of the more common methods, and it is simple to create with a very fast runtime. It is adequate for comparison purposes to paint an overall picture of the effectiveness of this thesis method.

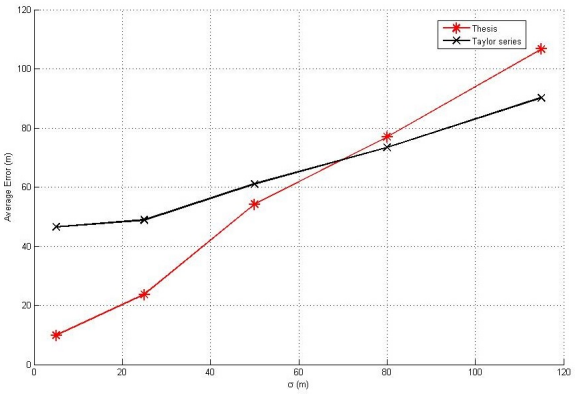

Figure 4.13: For each σ_N value, the simulation was run fifty times, and the average error from both methods is plotted. The red stars delineate the error values for this thesis method, and the black X's delineate the error values for the Taylor series method.

The next simulation compares geolocation results of the two methods using five σ_N values, the transmitter location $(75, 35)$, and the sensor configuration shown in Figure 4.11(a). Figure 4.13 plots the average error in the result of both methods across fifty simulations for each σ_N value. Table 4.4 lists these values and the average runtime for both methods. The offline runtime is still 17.45 minutes and is not included in the table.

Table 4.4: Comparison of σ_N values Averaged over 50 Simulations for Thesis (T) and Current Taylor Series (C) Methods

σ_N (m)	T Error (m)	C Error (m)	T Online (s)	C Runtime (s)
5	9.84	46.46	64.77	0.0091
25	23.73	48.88	68.25	0.0065
50	54.14	61.06	75.33	0.0065
80	76.94	73.36	78.82	0.0071
115	106.67	90.21	80.14	0.014

Table 4.5: Comparison of Sensor Configurations Averaged over 50 Simulations for Thesis (T) and Current Taylor Series (C) Methods

	Sensor Configuration, $\begin{matrix} x \\ y \end{matrix}$	T Error	C Error	T Offline	T Online	C Runtime
(a)	4 57 4 110 110 4 49 94 4 94	23.73 m	48.88 m	17.45 min	68.25 s	0.0065 s
(b)	20 38 56 74 92 60 60 60 60 60	115.52 m	NaN	17.81 min	73.85 s	0.013 s
(c)	93 103 108 103 93 10 30 50 70 90	85.62 m	NaN	17.40 min	58.46 s	0.011 s
(d)	2 42 42 2 22 8 8 48 48 28	86.52 m	165.60 m	17.57 min	127.16 s	0.0065 s
(e)	5 20 55 80 105 55 20 55 20 55	59.02 m	20.87 m	17.68 min	63.94 s	0.0054 s

The final simulation compares geolocation results of the two methods using five different sensor configurations, a σ_N of 25m, and the transmitter location $(75, 35)$. For visualization, one simulation for each sensor configuration is illustrated in Figure 4.14. The average values of the results are shown in Table 4.5. Since the offline runtimes differ slightly for each sensor configuration due to the differences in the *visible* and *visible_double* wall lists, the offline runtime for each configuration is included in the table.

The results presented in this chapter show that for a reasonable noise level, this thesis method performs more accurately than the current Taylor series method when obstructions are involved. However, in the cases when there are no obstructions, the thesis method is more susceptible to noise than the Taylor series method, which includes a noise correction

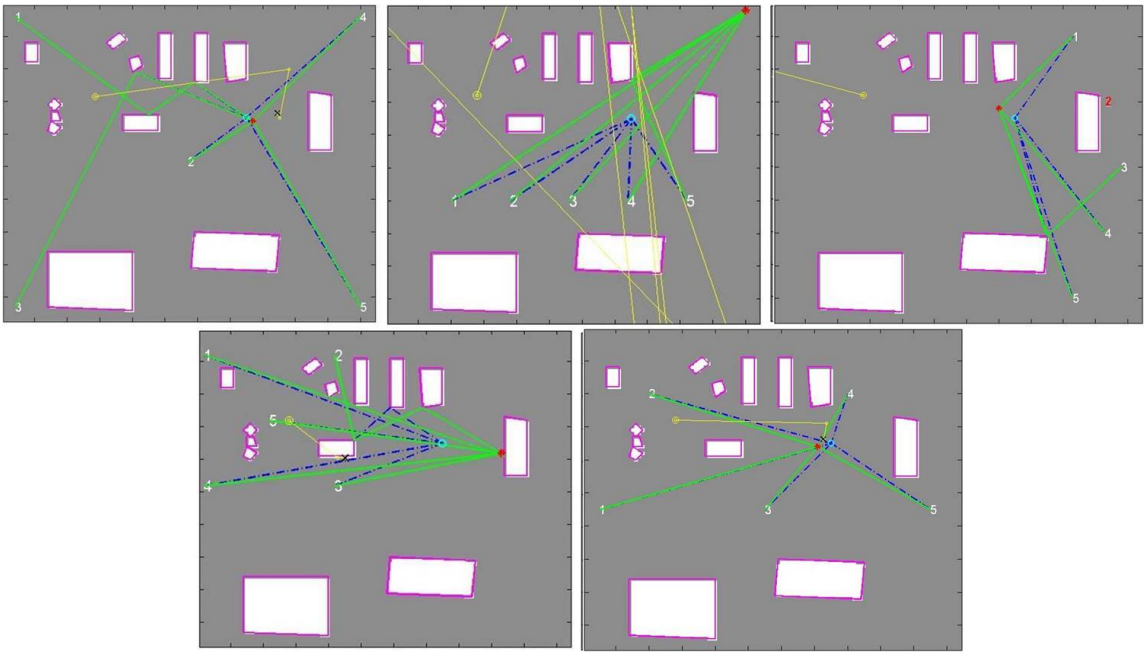

Figure 4.14: In each picture, the actual transmitter location is plotted as a cyan circle, and the actual shortest paths from the transmitter to the sensors are drawn as dashed blue lines. The result of this thesis algorithm is plotted as a red star, and the paths from this location to the sensors are drawn as green lines. Each iterative result of the current method is plotted as a yellow circle and connected to the next iterative result by a yellow line, in order to illustrate the trajectory of the iterations. The final result of the current method is plotted as a black X. Simulation results are shown from top left to bottom right for sensor configurations (a), (b), (c), (d), and (e), which are defined in Table 4.5. Original image data available from the U.S. Geological Survey [1].

when the σ_N is known, as it is here. Furthermore, there are obstruction situations when the error in the Taylor series result is mitigated. For example, transmitter location (e) in Table 4.3 causes spatially opposing sensors 3 and 4 to both measure inflated TDOA values. Since they are opposite one another, the inflation at these sensors essentially cancels each other out during the Taylor series calculations.

The results also show that sensor configuration (a) in Table 4.5 is the most ideal configuration for obtaining accurate results out of the five configurations which were simulated. Linear and near-linear sensor arrays, such as configurations (b) and (c), cause ambiguity in the results due to a loss in dimensionality. In a case like configuration (d), where the transmitter location is not within the field enclosed by the sensors, both methods appear to perform more poorly. For the Taylor series method, however, this is not a fair conclusion, since the the initial location guess caused the algorithm to terminate iterations at a local minimum value. However, choosing the initial location guess in order to avoid these situations is not trivial and is not explored in this thesis. Sensor configuration (d) may not be fairly represented either, since the situation simulated is a situation where there are no obstructions. Its performance may be comparable to that of configuration (a), but the comparison cannot be made since there are no potential transmitter locations which have an unobstructed path to all five sensors in configuration (a). The fact that it is so rare to find a completely unobstructed transmitter location proves how necessary it is to be able to account for the obstructions.

Unfortunately, the geolocation algorithm in this thesis takes much longer than the Taylor series method to calculate a result. The thesis method online times for the simulations in this chapter range from 20 seconds to 2 minutes. If the mission involves tracking a fast-moving target, then it may not be prudent to apply the thesis method for improved accuracy. Also, if the region of interest is an open field with no obstructions to the lines of sight of the sensors, then it does not make sense to use the thesis method.

V. Conclusions

THE goal of this research is to incorporate current image processing techniques with current geolocation techniques to improve transmitter geolocation when the transmitter location has an obstructed LOS to all of the ground sensors. Performance of the improved algorithm is compared to the performance of a current Taylor series geolocation method which does not consider obstructions. The comparison is made for several scenarios that vary in the image region, the noise standard deviation, the transmitter location, and the sensor configuration. Each scenario is simulated 50 times, and the performance is assessed via Monte Carlo analysis.

5.1 Summary

The resultant thesis algorithm can be divided into two main phases. The first phase is the building extraction algorithm, which is implemented as a compilation of existing techniques and not original. This algorithm takes an aerial, orthorectified, RGB image and determines building locations within the image. Estimates for minimum and maximum expected building perimeters are required inputs to the algorithm. The algorithm converts the image to grayscale intensity values and uses a threshold to determine which pixels belong to the shadows in the image. It is assumed that every building in the image casts a shadow. Therefore, the search for buildings begins with locating shadows. The shadow pixels are grouped into shadow objects which are evaluated for certain region properties which are likely to describe a shadow. Only the most likely shadow objects are retained. It is also assumed that every building casts a shadow in the same direction. Therefore, all shadows touch their respective buildings on the same side. The non-shadow pixels in the image are clustered and segmented into potential building objects based on their intensity values and connectivity. These objects are compared against the shadow objects. Those

which are adjacent to the shadow objects on the correct side are retained. The adjacent objects for a particular shadow are grouped into a single object, and this object is evaluated for region properties which are likely to describe a building. Finally, a best-fit rectangle is calculated for each object, under the assumption that buildings are rectangular. The rectangle of best fit for a particular building includes information about the building corner point locations and the normal vectors for each wall in the building.

After building locations are extracted, the thesis algorithm enters the second phase: geolocation. The geolocation algorithm is a simple grid search to find which location in the image provides theoretical TDOA values closest to the simulated actual TDOA values. This is not a new concept. However, the theoretical TDOA calculation method in this thesis improves upon current techniques. Theoretical TDOA values are calculated in the same way that the simulated actual TDOA values are calculated, except for the fact that noise is added for simulated actual TDOA calculation.

The TDOA values at the sensors are calculated from the theoretical TOA values at the sensors. The TOA value at each sensor is calculated using the image GSD, the speed of propagation of radio waves in air, and the distance the signal travels from the transmitter to that sensor. This distance, or path length, is where the image information comes into play. When a transmitter has an unobstructed LOS to a sensor, the shortest path between the two points is the LOS. However, with an obstructed LOS, finding the shortest path between the transmitter and the sensor is not this trivial. The TOA at a sensor for a given transmitter location cannot be predicted without knowing where the buildings are located in the image. The building extraction algorithm provides this knowledge. It is assumed that the signal can reflect off of any building wall and can only be reflected up to two times before its signal power is diminished to an undetectable level. It is also assumed that the building walls exhibit Lambertian reflectance, which simply means that a signal reflects in all directions from the side of the wall it hits regardless of the incidence angle. Based on

these assumptions in mind, the algorithm finds every possible single- and double-reflection path between the transmitter and the sensor. Then, the path with the shortest length is used to determine the TOA at that sensor.

For simulated actual values, noise is added to the TOA at each sensor before calculating the TDOA at each sensor. For a grid search guess, no noise is added. When comparing the sensor set of TDOA values for a grid search guess to the simulated actual set of TDOA values, a measurement referred to in this thesis as $diff$ is calculated. The $diff$ for a particular grid search guess is the average across the sensors of the square of the difference between the theoretical TDOA value for a sensor and that sensor's simulated actual TDOA value. The grid search guess yielding the smallest $diff$ is the geolocation estimate.

Results of the thesis algorithm are compared with the performance achieved using a current geolocation technique. The current technique uses the Taylor series method of solving for the intersection of the hyperbolic curves created by the measured TDOA values.

5.2 Impact

When compared with the Taylor series method, the thesis method improves the geolocation error by an average of 44m, or 53% in the obstructed simulation cases. This improvement is based on the 25m σ_N simulations and does not include the cases in which the Taylor series method did not converge or the case in which the Taylor series result was a local minimum.

Each simulation was run 50 times to find the average error and online runtime for both methods. The runtime of the thesis algorithm is divided into an offline time and an online time. The offline time includes the time it takes to apply the building extraction algorithm to an image and the time it takes to populate the *visible* and *visible_double* wall lists which describe portions of potential signal paths. These offline operations only need to be

performed once for a given image region and sensor configuration. The offline runtimes vary significantly from one image to the next based mostly on the original size of the image and on the number of buildings and shadows in the image. The offline runtimes for the simulations included here range from about 7 minutes to about 30 minutes. The online runtime of the thesis algorithm includes the time it takes to complete the path portions from *visible* and *visible_double* at each grid search guess, to calculate the TDOA values for each grid guess, and to compare these values to the simulated actual values.

The average online runtimes for the included simulations range from around 20 seconds to around 2 minutes. This is significantly slower than the 0.02 average runtime of the Taylor-series algorithm which is computed entirely online.

Depending on the mission, the slower computation time of the thesis algorithm may be tolerable given the improvement in geolocation estimation. Furthermore, the offline runtime may be a non-issue, since the offline portions can be performed before the start of the mission. For example, if the mission involves tracking a target in a particular region with stationary sensors, geolocation can be performed any number of times without having to recompute the offline portions.

Another drawback of the thesis method is that it does not include a noise correction capability. Therefore, for situations when all of the sensors have an unobstructed LOS, the thesis method performs less accurately than the Taylor series method. However, there may be a way to use the difference between measured and expected signal strengths to determine whether the measured TDOA is based on a reflection path or from a direct path. Whether or not obstructions exist can dictate which geolocation method is used.

Another potential drawback of the thesis method is that it cannot find the location of a target that is outside of the image area. This issue can be overcome simply by padding the search area. There will not be building information available for the padded area, but even knowing some of the buildings in a region is helpful. As shown by sensor configuration

(d) in Figure 4.14, however, the algorithm tends to be less accurate when the transmitter is not located within the sensor region.

5.3 Recommendations for Future Work

The algorithm developed under this research represents a conceptual foundation for including image information in the transmitter geolocation process. There are many ways in which the algorithm can be improved. First, it is important to assess the algorithm's performance with experimental data, rather than simulated data. Second, this thesis assumed that all buildings are tall enough to reflect or obstruct the signal and that all sensors and transmitters are far enough off the ground such that the ground does not interfere with the signal. These assumptions essentially create a 2-dimensional geolocation plane. It would be more accurate to treat the problem as 3-dimensional, which would require more advanced image processing to determine the height of the buildings.

Another area for improvement is the unobstructed case. As noted previously, there may be a way to classify the TDOA as either reflected or direct based on measured and expected signal strength. It would also be helpful to develop some form of noise correction capability.

The biggest drawback to the thesis algorithm is the amount of required computation time. Unfortunately, any type of grid search will be time consuming. It is therefore desirable to eliminate the need for a grid search. There may be other existing geolocation techniques that could be adapted to include the image intelligence.

Bibliography

[1] "U.S. Department of the Interior U.S. Geological Survey", 11/22/2013. URL http://earthexplorer.usgs.gov.

[2] Bhatt, A. D., U. Gupta, V. Wagholikar, and U. V. Pise. "Edge Detection and Segmentation of Multiple Contours from CT Scan Images". *Computer-Aided Design Applications*, 9(4):501–516, 07 2012. URL http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=83769010&site=ehost-live.

[3] Chan, Y. T. and K. C. Ho. "A simple and efficient estimator for hyperbolic location". *IEEE Transactions on Signal Processing*, 42(8):1905–1915, 1994.

[4] Duda, R., P. Hart, and D. Stork. *Pattern Classification*. Academic Internet Publishers, 2 edition, 2006.

[5] Kay, S. M. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice-Hall, 1993.

[6] Lin, C. and R. Nevatia. "Building Detection and Description from a Single Intensity Image", 1998.

[7] Mathews, K. "Radio Paths with Obstructions: The Shortest Radio-Path Algorithm", AFIT Technical Report, 2013.

[8] Mueller, S. and D. W. Zaum. "Robust Building Detection in Aerial Images". *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVI:143–148, 2005.

[9] Pal, N. R. and S. K. Pal. "A review on image segmentation techniques". *Pattern Recognition*, 26(9):1277–1294, 9 1993.

[10] Reza, R. I. *Data fusion for improved TOA/TDOA position determination in wireless systems*. Master of science in electrical engineering, Virginia Polytechnic Institute and State University, 2000.

[11] Sohn, G. and I. J. Dowman. "Extraction of buildings from high resolution satellite data". *Automated Extraction of Man-Made Objects from Aerial and Space Images (III). Balkema Publishers*, 345–355. Balkema Publishers, 2001.

[12] Strobel, N. and R. Rabenstein. "Classification of time delay estimates for robust speaker localization". *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, 3081–3084. 1999.

[13] Tsai, V. J. D. "A comparative study on shadow compensation of color aerial images in invariant color models". *IEEE Transactions on Geoscience and Remote Sensing*, 44(6):1661–1671, 2006.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27–03–2014	Master's Thesis	Oct 2012–Mar 2014

4. TITLE AND SUBTITLE

Combining Image Processing with Signal Processing to Improve Transmitter Geolocation Estimation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Abraham, Amy M., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-14-M-01

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A.
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research develops an algorithm which combines image processing with signal processing to improve transmitter geolocation capability. A building extraction algorithm is compiled from current techniques in order to provide the locations of rectangular buildings within an aerial, orthorectified, RGB image to a geolocation algorithm. The geolocation algorithm relies on measured TDOA data from multiple ground sensors to locate a transmitter by searching a grid of possible transmitter locations within the image region. At each evaluated grid point, theoretical TDOA values are computed for comparison to the measured TDOA values. To compute the theoretical values, the shortest path length between the transmitter and each of the sensors is determined. The building locations are used to determine if the LOS path between these two points is obstructed and what would be the shortest reflected path length. The grid location producing theoretical TDOA values closest to the measured TDOA values is the result of the algorithm. Measured TDOA data is simulated in this thesis. The thesis method performance is compared to that of a current geolocation method that uses Taylor series expansion to solve for the intersection of hyperbolic curves created by the TDOA data. The average online runtime of thesis simulations range from around 20 seconds to around 2 minutes, while the Taylor series method only takes about 0.02 seconds. The thesis method also includes an offline runtime of up to 30 minutes for a given image region and sensor configuration. The thesis method improves transmitter geolocation error by an average of 44m, or 53% in the obstructed simulation cases when compared with the current Taylor series method. However, in cases when all sensors have a direct LOS, the current method performs more accurately. Therefore, the thesis method is most applicable to missions requiring tracking of slower-moving targets in an urban environment with stationary sensors.

15. SUBJECT TERMS

Geolocation, Image Segmentation, Time Difference of Arrival

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Richard K. Martin (ENG)
U	U	U	UU	106	19b. TELEPHONE NUMBER *(include area code)* (937) 2553636 x4625; Richard.Martin@afit.edu